

CRYPTOGRAPHY

(Lecture 1)

Lecturer: Elena Pagnin - Year: 2024

Lecture Agenda

Introduction

- Cryptography: Meaning and Aims
- Brief History
- Kerckhoffs's Principle
- Foundations of Modern Cryptography

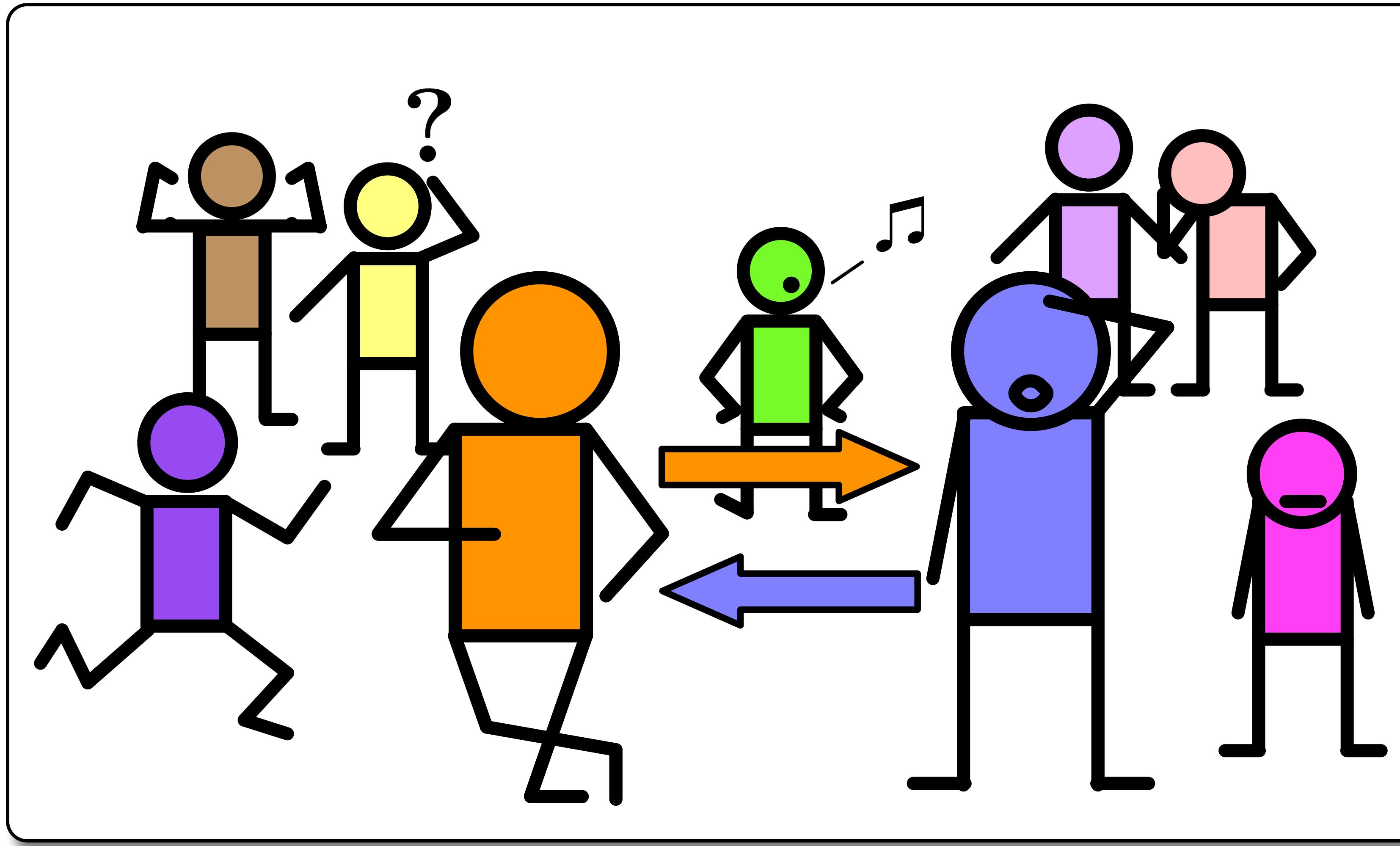
Symmetric Encryption

- Secure Communication
- Symmetric Encryption
- The One Time Pad (OTP) [Proof]
- Perfect Secrecy
- Shannon's Theorem [Proof]

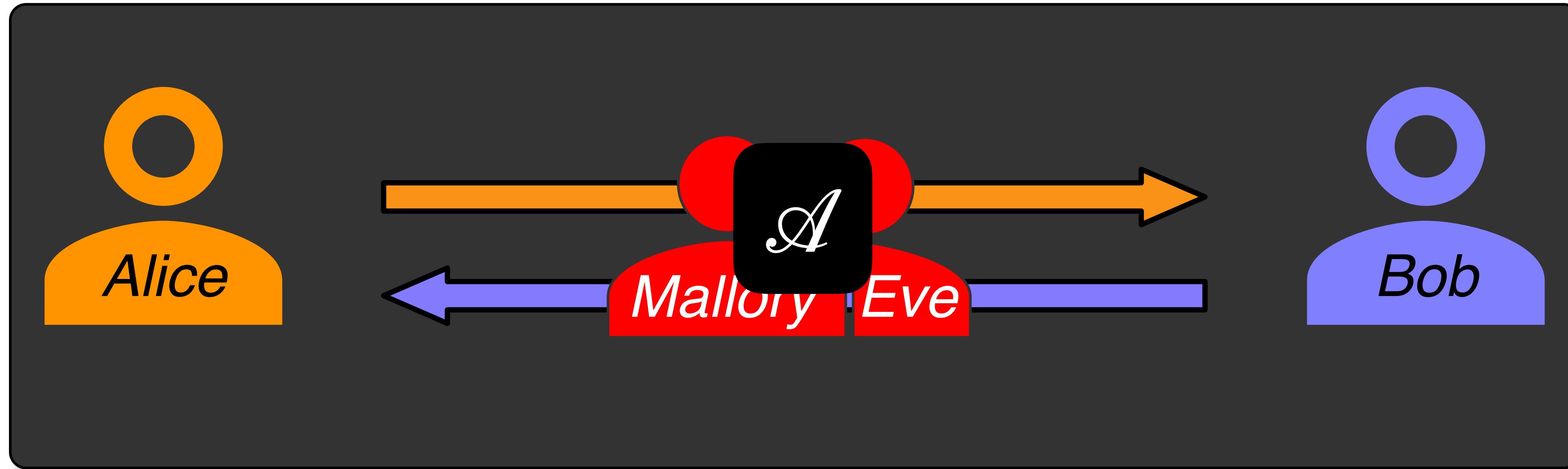
Pseudorandom Generators (PRG)

- Definition
-

The Real World



The Dark World of Cryptography (and Cryptographers)



The Goal of Cryptography: “Make our Digital World Safe”

- 🔒 Confidentiality
- ✉️ Data integrity
- ✍️ Authenticity
- >ID Entity identification

- 🚫 Access control / authorisation
- 🎭 Anonymity
- ✓ Non-repudiation
- 👁️ Privacy

Brief History



Gaius Julius Caesar



Mary Queen of Scots

Who?



The Enigma Machine (Mavis Batey)



Gene Grabeel

Common issues:

- No rigorous security estimates / analysis
- “Security by Obscurity”
- Bad security “hygiene”

Kerckhoffs's Principle

Kerckhoffs's Principle (1883)

A cryptographic system should be secure even if everything about the system, except for the key, is public knowledge.

In Claude Shannon's words: **the enemy knows the system**, so we should design systems under the assumption that the enemy will immediately gain full familiarity with them (no security via obscurity)

Foundations of *Modern* Cryptography (1980-Now)

CRYPTOGRAPHY

CRYPTANALYSIS

CRYPTOLOGY

Rigorous definitions

- What does security mean?
- What are the attacker's goal and resources
- Precise mathematical security assumptions
 - (formally define “hard”)

Rigorous logical reasoning to prove security

Lots of heuristics to define exact security levels

Solutions need to work in practice

- Efficient algorithms
- Use the lowest ratio $\frac{\text{size}}{\text{security}}$

When I say “crypto” I mean “cryptography” not “cryptocurrency”

Provable Security

- Providing mathematical *definitions of security* (e.g. through “security games” between algorithms and probabilities to quantify the chance of “winning”)
- Providing *precise mathematical assumptions* (e.g. “the discrete logarithm problem is *hard*”, where *hard* is formally defined).
- Providing *proofs of security* (often reasoning by **reduction**: if a scheme can be broken, then there is a way to contradict a mathematical assumption. Hence, *if we believe in the mathematical assumption, the scheme cannot be broken*).



Lecture Agenda

Introduction

- Cryptography: Meaning and Aims
- Brief History
- Kerckhoffs's Principle
- Foundations of Modern Cryptography

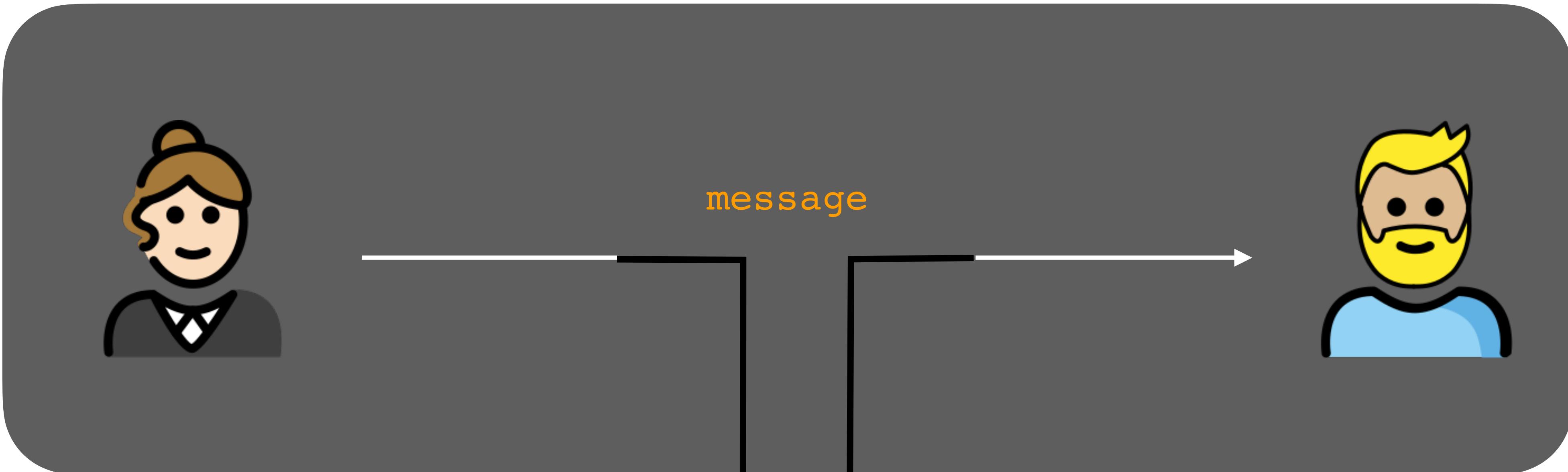
Symmetric Encryption

- Secure Communication
- Symmetric Encryption
- The One Time Pad (OTP) [Proof]
- Perfect Secrecy
- Shannon's Theorem [Proof]

Pseudorandom Generators (PRG)

- Definition
-

Secure Communication Over an Insecure Channel



- What's \mathcal{A} 's *goal*?
- What are \mathcal{A} 's *resources*?
- What does *secure* mean?
- What can Alice and Bob **use**?

- *Learn the message*
- *Eavesdrop the channel*
- *...we will find out...*
- *Symmetric encryption*

Useful Terminology

Efficient Algorithm : an algorithm that runs in time polynomial in its input (fast).

Symmetric Encryption aka Private-Key Encryption (or simply encryption).

Scheme : a collection of algorithms that together satisfy a number of properties.

Protocol : a scheme where algorithms usually interact with one another.

More Terminology and Notation

Deterministic : refers to a value that is set, or to a function that given an input always returns the same output.

Notation: $b = 0$, $\text{Alg}(x) = y$

Random : refers to a value that is drawn from a set using the uniform distribution (all possibilities are equiprobable).

Notation: $b \leftarrow \$\{0,1\}$

Randomised or **Probabilistic** : refers to a function or algorithm that involves sampling and using randomness, thus the output is non-deterministic (unless the randomness is specified).

Notation: $y \leftarrow \text{Alg}(x)$ and there exists $rnd \in \{0,1\}^n$ such that $y = \text{Alg}(x; rnd)$

Conditional Probability : $Pr[A | B]$ aka $Pr[B : A]$

Symmetric Encryption - Syntax

Definition: Symmetric Encryption

A tuple $(KeyGen, E, D)$ is a symmetric encryption scheme over the sets \mathcal{K} (key space), \mathcal{M} (message space), and \mathcal{C} (ciphertext space) if all algorithms are efficient and satisfy the following:

$KeyGen(1^n) \rightarrow k$: the key generation is a randomised algorithm that returns a key k . (This algorithm is often implicit when $k \leftarrow \$\mathcal{K}$)

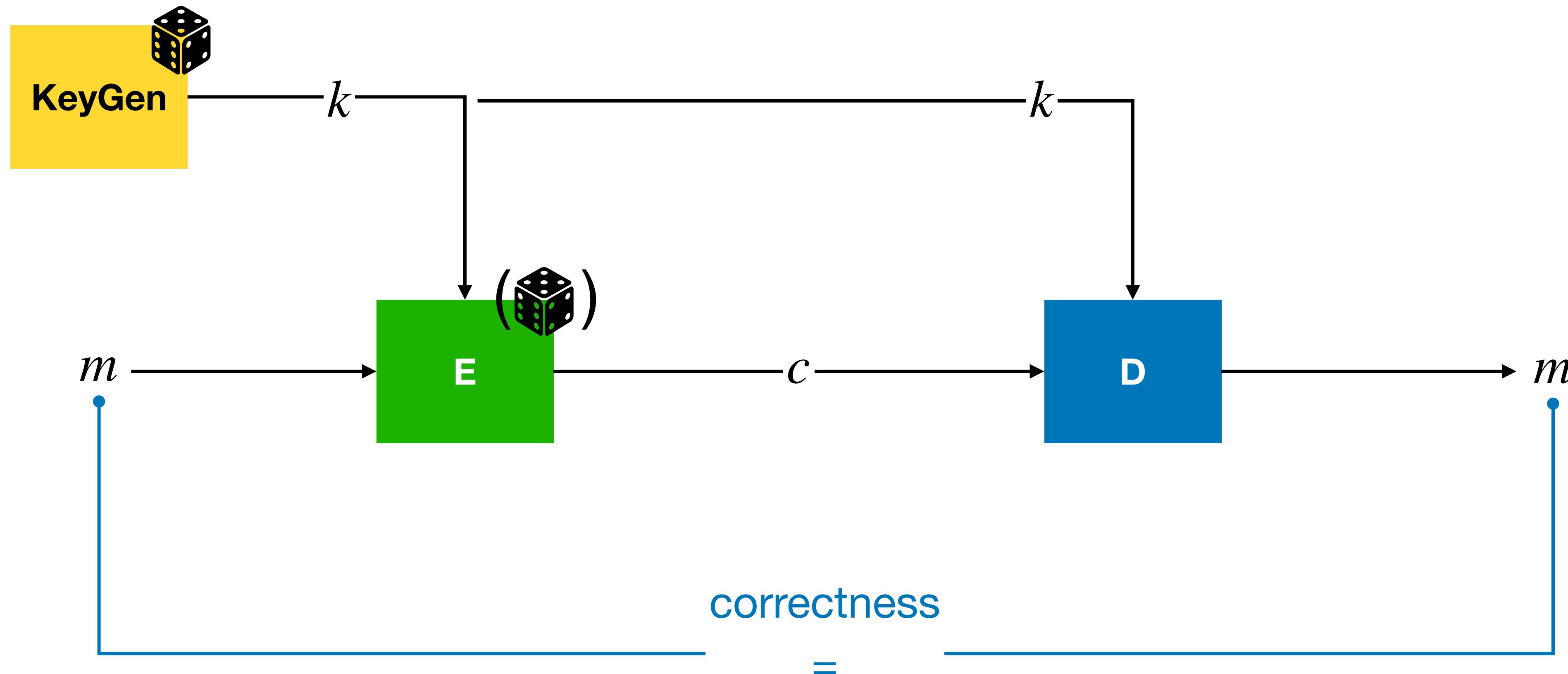
$E(k, m) \rightarrow c$: the encryption is a possibly randomised algorithm that on input a key k and a (plaintext) message m , outputs a ciphertext c .

$D(k, c) \rightarrow m$: the decryption is a deterministic algorithm that on input a key k and ciphertext c , outputs a plaintext message m .

CORRECTNESS:

$$\Pr[D(k, E(k, m)) = m \mid k \leftarrow KeyGen(1^n)] = 1 \quad \dots \text{for all messages } m \in \mathcal{M}$$

Symmetric Encryption - Visualisation



Symmetric Encryption - the One Time Pad (OTP)

Definition: Symmetric Encryption

A tuple $(KeyGen, E, D)$ is a symmetric encryption scheme if \mathcal{K} (key space), \mathcal{M} (message space) and \mathcal{C} (ciphertext space) are efficiently samplable, the encryption and decryption algorithms are efficient and satisfy the following properties:

$KeyGen(1^n) \rightarrow k$: the key generation algorithm takes a security parameter 1^n and returns a key k . (This algorithm is often called a key generator)

$E(k, m) \rightarrow c$: the encryption is a polynomial time algorithm that takes as input a key k and a (plaintext) message m and outputs a ciphertext c .

$D(k, c) \rightarrow m$: the decryption is a polynomial time algorithm that takes as input a key k and ciphertext c , outputs a plaintext m .

$D(k, E(k, m)) = m$ for all messages $m \in \mathcal{M}$

CORRECTNESS:

$$\Pr[D(k, E(k, m)) = m \mid k \leftarrow KeyGen(1^n)] = 1 \text{ for all messages } m \in \mathcal{M}$$

Example: the One Time Pad (OTP)

$$\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^n$$

$$KeyGen(1^n) \rightarrow k \text{ (where } k \leftarrow \{0,1\}^n)$$

$$E(k, m) = k \oplus m$$

$$D(k, c) = k \oplus c$$

Key: 1 0 1 1 0 0 1 1 1 1 0 0 1

\oplus

Plaintext: 0 1 1 0 1 0 0 0 1 1 0 1

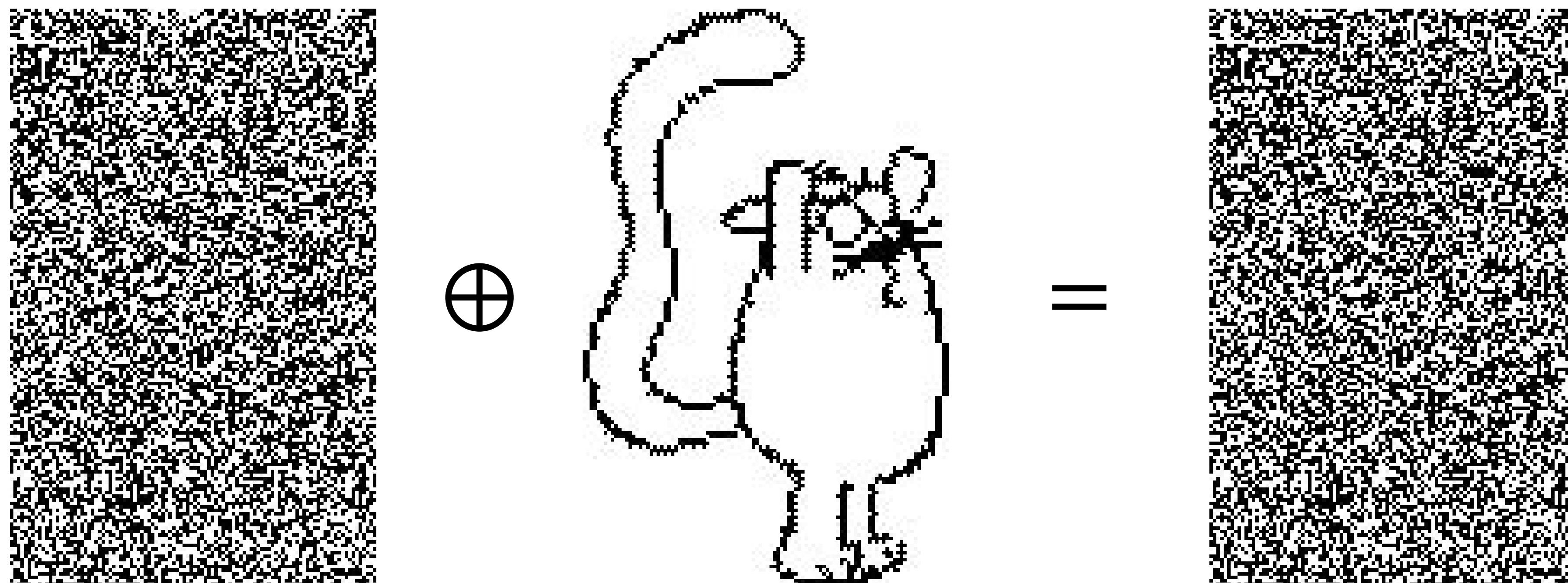
Ciphertext: 1 1 0 1 1 0 1 1 0 1 0 0

OTP From the Attacker's Point of View

Uniformly
random
key

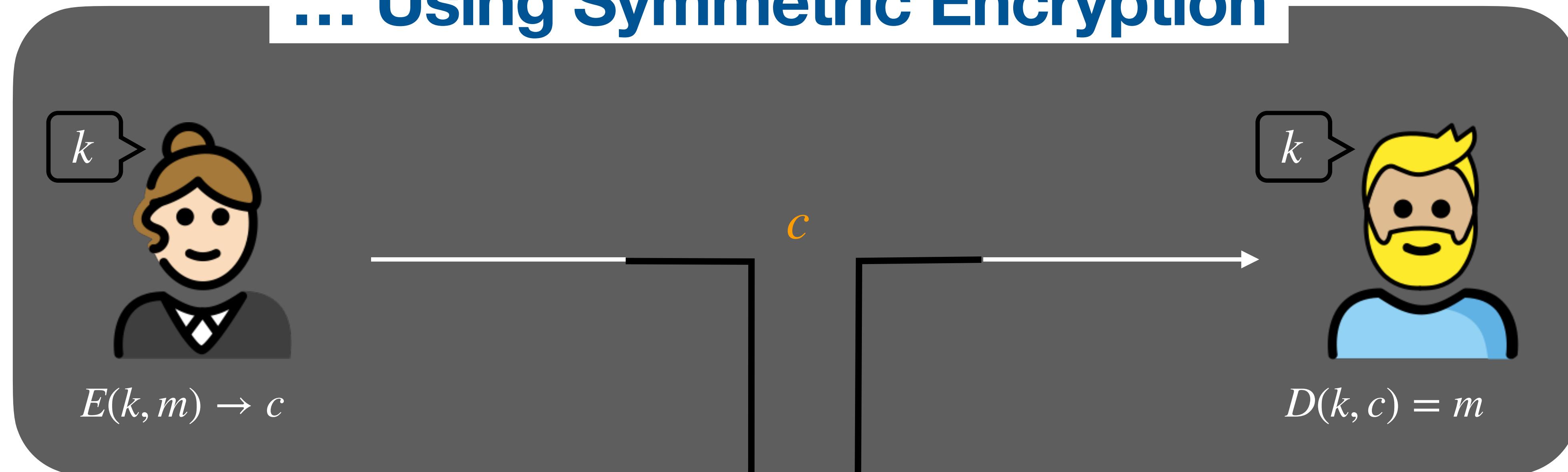
$$k \oplus m = c$$

Uniformly
random
ciphertext



Secure Communication Over an Insecure Channel

... Using Symmetric Encryption



■ 🤔 What does **secure** mean?

“ \mathcal{A} should not learn the message”

“The ciphertext c should **not leak any information** about the message m ”
(a part from the length of m)

Perfect Secrecy

Definition: Perfect Secrecy (Perfect Security)

A symmetric encryption scheme $(KeyGen, E, D)$ is perfectly secret if for all pair of messages $m_0, m_1 \in \mathcal{M}$ and for all ciphertexts c it holds that:

$$Pr[c \leftarrow E(k, m_0) | k \leftarrow KeyGen(1^n)] = Pr[c \leftarrow E(k, m_1) | k \leftarrow KeyGen(1^n)]$$

This security notions essentially states that:

For every pair of messages, the probabilities that either message maps to a given ciphertext c must be equal.

The First Security Proof in This Course

The One-Time Pad is a perfectly secure private-key encryption scheme.

Proof:

First, show that OTP is a symmetric encryption scheme.

To do so, define $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^n$, and the algorithms

$\text{KeyGen}(1^n) : k \leftarrow \$\{0,1\}^n$, $E(k, m) = k \oplus m$, and $D(k, c) = k \oplus c$.

Correctness holds since for all k , and all m : $D(k, E(k, m)) = k \oplus c = k \oplus (k \oplus m) = m$.

Next, notice that for every message m and ciphertext c there is exactly one key $k \in \{0,1\}^n$ such that $c = E(k, m)$ (the key is $k = m \oplus c$). Then:

$$\Pr[c \leftarrow E(k, m) \mid k \leftarrow \text{KeyGen}(1^n)] = \Pr[c \oplus m \leftarrow \text{KeyGen}(1^n)] = \frac{1}{|\mathcal{K}|}$$

Since this holds for every message m , we have that:

$$\Pr[E(k, m_0) \rightarrow c \mid k \leftarrow \text{KeyGen}(1^n)] = \frac{1}{|\mathcal{K}|} = \Pr[E(k, m_1) \rightarrow c \mid k \leftarrow \text{KeyGen}(1^n)]$$

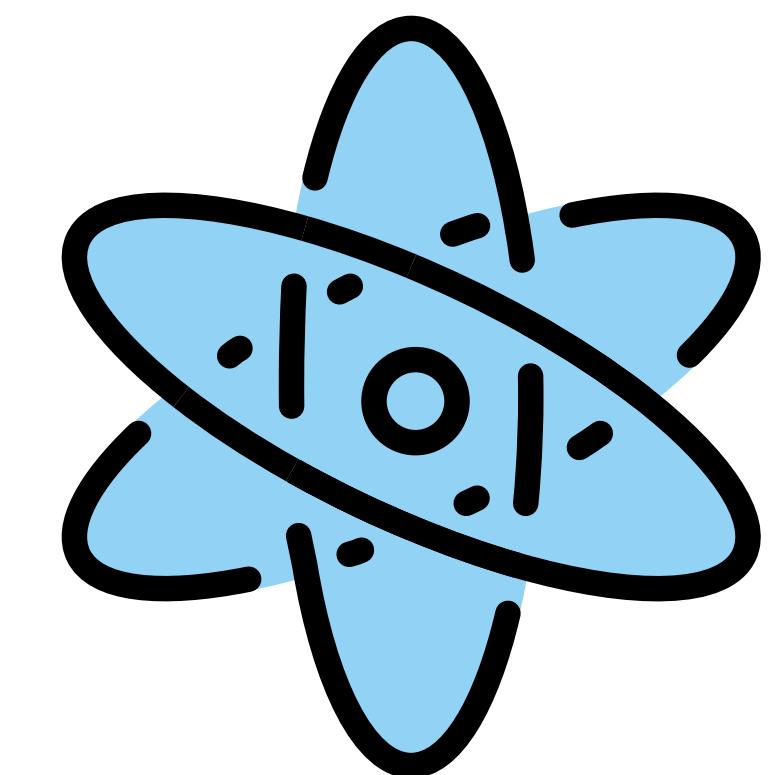
Security: Information-Theoretic VS Computational

Perfect-secrecy is a form of **information-theoretic** (aka unconditional) security notion, that means: the security property holds *no matter what* computational power, time-, or space-limit the adversary has.

Computational security is based on the *conjectured gap* between the complexity classes P and NP, that means: we have efficient algorithms for the legitimate user, and inefficient algorithm for the adversary.

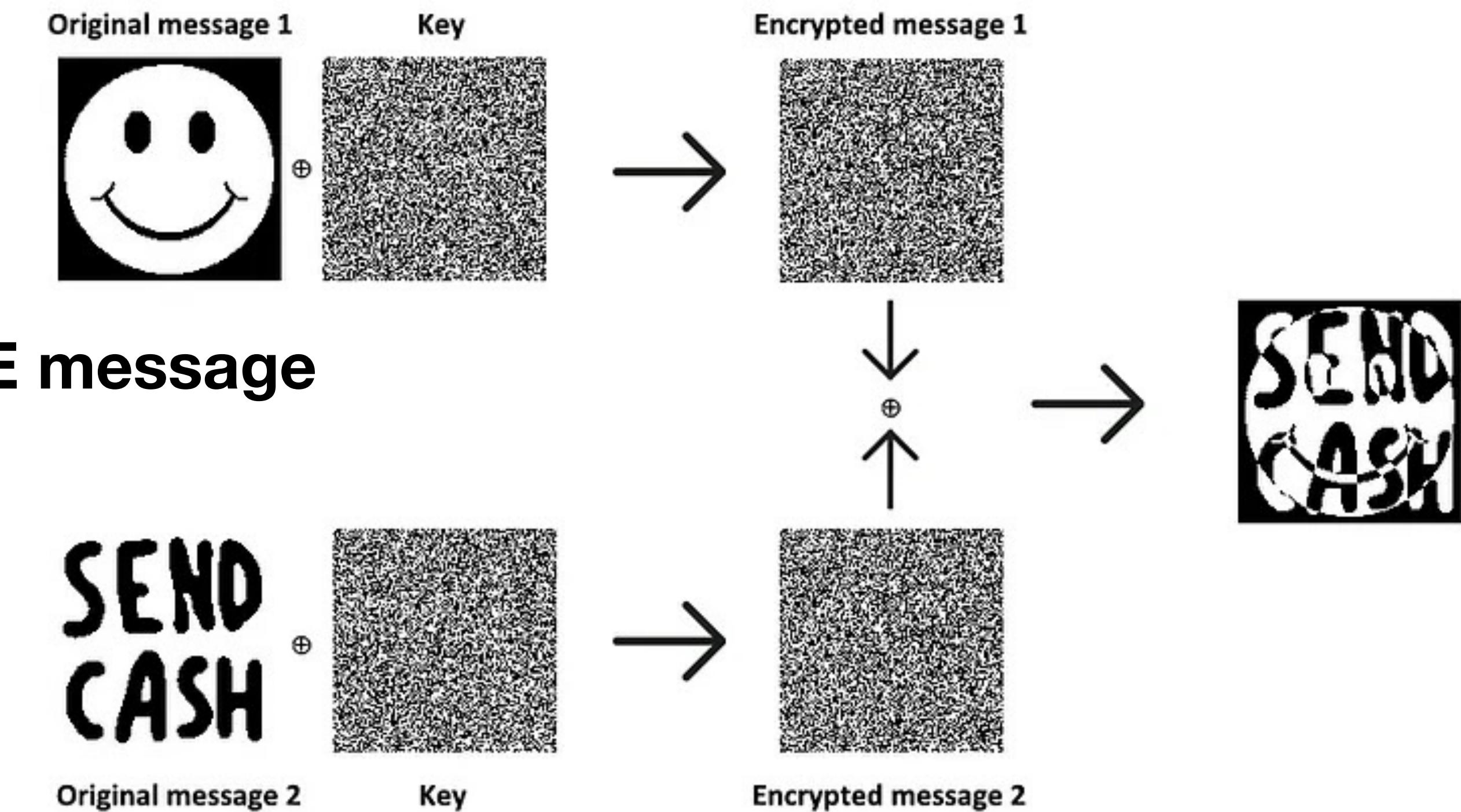
A **gap** is a hard problem (can be verified quickly but no efficient algorithm is known to solve it). What is *computationally hard* depends on the adversary's resources (classical vs quantum computing, storage, time..)

Computational solutions are usually more efficient and practical than information-theoretic ones **BUT** they always have a “best before date” (we'll discuss this during the course)



One Time Pad: Problems

1- The key is as long as the message



Assume that the same key is used twice to produce $c_0 = k \oplus m_0$ and $c_1 = k \oplus m_1$.

Assume the adversary \mathcal{A} gets hold of the two ciphertexts, then \mathcal{A} can compute:

$$c_0 \oplus c_1 = (k \oplus m_0) \oplus (k \oplus m_1) = m_0 \oplus m_1$$

But $m_0 \oplus m_1$ conveys a lot of information about m_0 and m_1 . This is not acceptable.

3- The ciphertext is (intentionally!) malleable



Shannon's Theorem

Theorem (Shannon 1940s)

If a symmetric encryption scheme (KeyGen, E, D) defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ has perfect security then $|\mathcal{K}| \geq |\mathcal{M}|$.

Proof: Fix arbitrary $m_0 \in \mathcal{M}$ and $k_0 \in \mathcal{K}$, and let $c_0 = E(k_0, m_0)$.

Since the encryption scheme has perfect secrecy, it holds that “every message has the same probability to be encrypted to c ”, i.e., $\Pr[c_0 = E(k, m)] = \Pr[c_0 = E(k, m_0)] > 0$.

Thus for each $m_i \in \mathcal{M}$ there is a key $k_i \in \mathcal{K}$ such that $E(k_i, m_i) = c_0$.

But each message must have a different key! If the same key k would map two plaintexts m_1 and m_0 to the same ciphertext $c_0 = E(k, m_1) = E(k, m_0)$, then we lose correctness (the decryption of c_0 for the key k becomes ambiguous).

Thus there must be at least as many keys as messages: $|\mathcal{K}| \geq |\mathcal{M}|$.

Take away: **perfect security is impractical**



**How close to perfect security can
we go, while being practical?**

A Little Secret: the Core of Crypto Is Randomness



This is the
goal of
**Pseudo
Random
Generators
(PRG)**

The perfect secrecy of OTP comes from using one random key to mask/hide one message. Since we cannot reuse the key (otherwise we lose security), can we ‘expand’ it?

Lecture Agenda

Introduction

- Cryptography: Meaning and Aims
- Brief History
- Kerckhoffs's Principle
- Foundations of Modern Cryptography

Symmetric Encryption

- Secure Communication
- Symmetric Encryption
- The One Time Pad (OTP) [Proof]
- Perfect Secrecy
- Shannon's Theorem [Proof]

Pseudorandom Generators (PRG)

- Definition
-

Pseudo Random Generators (PRG)

Definition: PRG

A Pseudo Random Generator is a deterministic function $\text{PRG} : \{0,1\}^n \rightarrow \{0,1\}^\ell$ such that:

- $\text{PRG}(\cdot)$ is efficiently computable
- $\text{PRG}(\cdot)$ expands its input, i.e., $\ell > n$
- The ensemble $\{\text{PRG}(x) \mid x \leftarrow \{0,1\}^n\}$ is *pseudo-random*, i.e., for $x \leftarrow \{0,1\}^n$ no efficient adversary can tell apart $\text{PRG}(x)$ from a random string $y \leftarrow \{0,1\}^\ell$.

The best way to check if a candidate algorithm is a PRG is by running a series of tests, there is no mathematical proof! But we can reason about the *security* of a PRG using a formal (mathematical) security game.

Recap From the Last Lecture

Kerckhoff's Principle

Some definitions, notations, and important terminology

Symmetric Encryption (and **CORRECTNESS**)

Perfect Secrecy

The One Time Pad (OTP). $c = k \oplus m$

Shannon's Theorem **PROOF**

perfect security $\Rightarrow |\mathcal{K}| \geq |\mathcal{M}|$

PROOF

The OTP is perfectly secure

How close to perfect security can we go, while being practical?

Lecture 2 - Agenda

Pseudorandom Generators (PRG)

- Definition
- Security (Real or Random?)
- Negligible
- The RSA PRG
- Secure encryption from PRG
- Semantic Security (Left or Right)
- PRG-Based Encryption [Proof]

Block Ciphers (Continued)

- Advanced Encryption Standard
- Design Principles

Modes of Operation

- ECB, CBC, CTR
- Is AES-ECB Semantically Secure?

Block Ciphers

- Pseudo Random Permutations
- Definition of Block Cipher
- Chosen Plaintext Attack (IND-CPA)

Pseudo Random Generators (PRG)

Definition: PRG

A Pseudo Random Generator is a deterministic function $\text{PRG} : \{0,1\}^n \rightarrow \{0,1\}^\ell$ such that:

- $\text{PRG}(\cdot)$ is efficiently computable
- $\text{PRG}(\cdot)$ expands its input, i.e., $\ell > n$
- The ensemble $\{\text{PRG}(x) \mid x \leftarrow \{0,1\}^n\}$ is *pseudo-random*, i.e., for $x \leftarrow \{0,1\}^n$ no efficient adversary can tell apart $\text{PRG}(x)$ from a random string $y \leftarrow \{0,1\}^\ell$.

The best way to check if a candidate algorithm is a PRG is by running a series of tests, there is no mathematical proof! But we can reason about the *security* of a PRG using a formal (mathematical) security game.

Example: RSA PRG

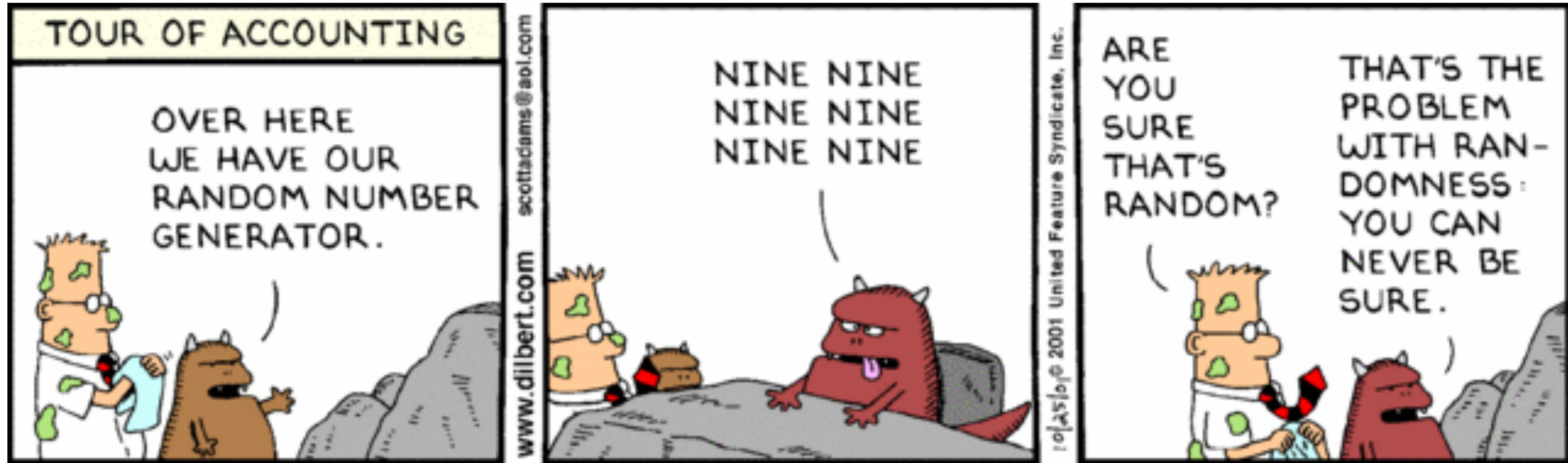
RSA PRG

Let p, q be two n -bit long primes, $N = p \cdot q$, and $e \in \mathbb{Z}_N^*$ a random invertible element. Let $LSB(x)$ be the function that returns the least significant bit of its input $x \in \mathbb{Z}_N$. Then $\text{PRG}_{RSA} : \{0,1\}^n \rightarrow \{0,1\}^\ell$ is defined as

$$\text{PRG}_{RSA}(x) = LSB(x) \parallel LSB(x^e \bmod N) \parallel \cdots \parallel LSB(x^{e^\ell})$$

```
N = 531761
e = 493589
x = 72979
PRG_RSA(x) = [1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0]
```

Pseudo Random Generators (PRG)



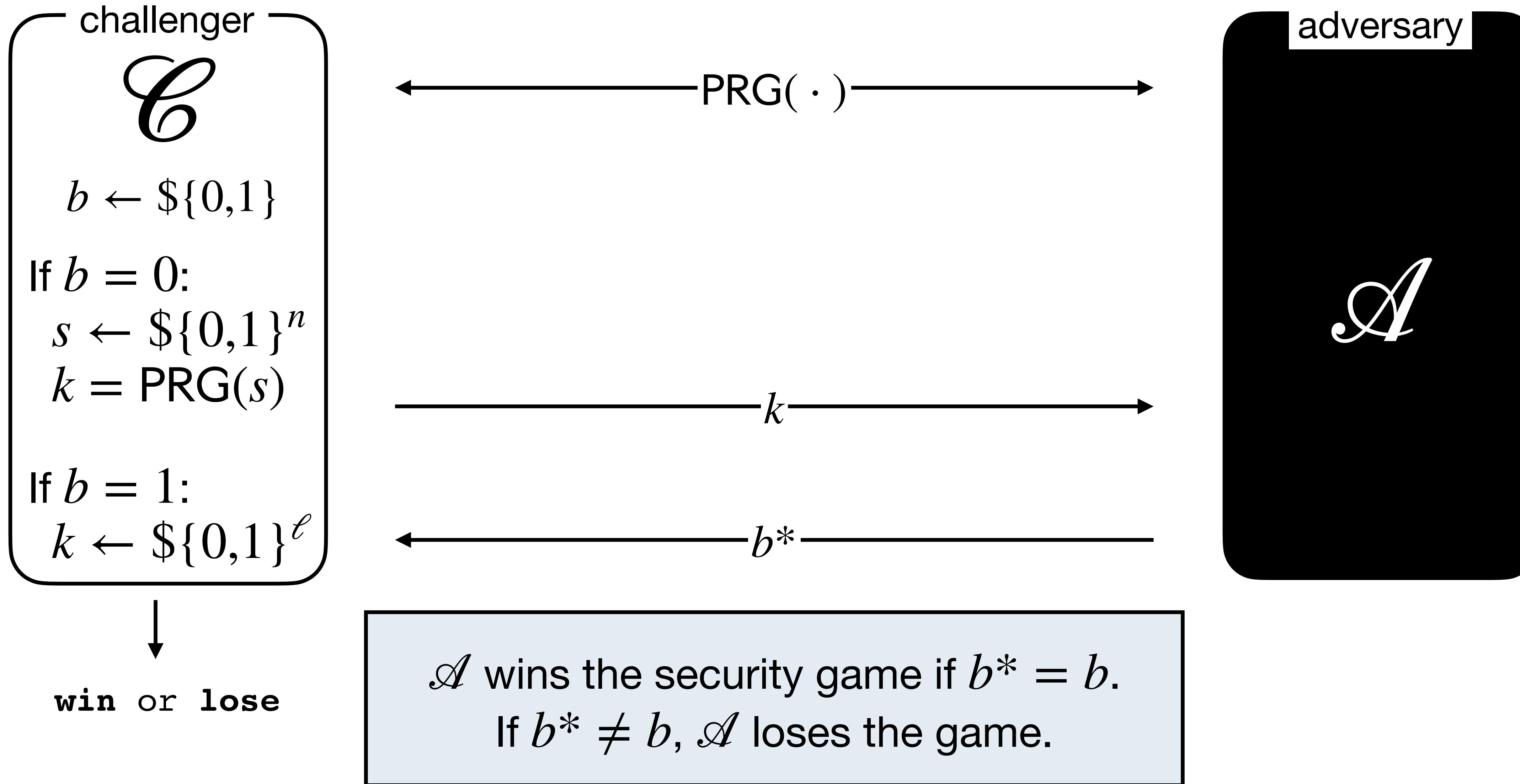
The best way to check if a candidate algorithm is a PRG is by running a series of tests, there is no mathematical proof! **But we can reason about the security of a PRG using a formal (mathematical) security game.**

“Real OR Random” Security (Intuition)



Security Game For PRG

Aim: quantify the attacker's likelihood in distinguishing PRG from a source of uniform randomness over $\{0,1\}^\ell$



Security Game For PRG

Definition: Secure PRG

A pseudo random function $\text{PRG} : \{0,1\}^n \rightarrow \{0,1\}^\ell$ is a secure PRG if any probabilistic polynomial time attacker \mathcal{A} has only negligible advantage in winning the PRG security game. Formally,

$$\text{Adv}(\mathcal{A}) = \left| \Pr[\mathcal{A} \text{ wins}] - \frac{1}{2} \right| < \text{negl}(n)$$

Verbose description of the PRG security game

1. The challenger \mathcal{C} draws a uniformly random bit $b \leftarrow \$\{0,1\}$.
2. If $b = 0$, the challenger draws a random seed $s \leftarrow \$\{0,1\}^n$ and computes $k = \text{PRG}(s)$.
If $b = 1$, the challenger draws a uniformly random string $k \leftarrow \$\{0,1\}^\ell$.
3. \mathcal{C} sends k to \mathcal{A} .
4. \mathcal{A} tries to determine b from k , and eventually (within polynomial time) returns its guess b^* .
5. \mathcal{A} sends b^* to the \mathcal{C} . The adversary wins if $b^* = b$.

What Does Negligible Mean?

"too small to matter"

Negligible A function $negl(x) : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is **negligible** in x if for any positive polynomial $p(x)$ it holds that

$$negl(x) \leq \frac{1}{p(x)} \quad \text{for all } x \geq x_0 \in \mathbb{N}$$

Intuition: Events that occur with negligible probability occur so seldom that polynomial time algorithms will never see them happening.

This definition is asymptotic ("it holds from a certain point onwards"). This is a common approach in complexity-based cryptography.

In practice, if one needs to pick a value, then $negl(x) < 2^{-128}$ is considered to be negligible (but this depends on the context, and may yield inefficient constructions).

Different Formulation, Same Meaning

intensivecrypto.org

Definition 3.2 (Pseudorandom generator)

Let $G : \{0, 1\}^* \rightarrow \{0, 1\}^*$ be some function computable in polynomial time. We say that G is a *pseudorandom generator* with length function $\ell : \mathbb{N} \rightarrow \mathbb{N}$ (where $\ell(n) > n$) if

- For every $x \in \{0, 1\}^*$, $|G(x)| = \ell(|x|)$.
- For every polynomial $p(\cdot)$ and sufficiently large n , if $D : \{0, 1\}^{\ell(n)} \rightarrow \{0, 1\}$ is computable by at most $p(n)$ operations, then

$$|\Pr[D(G(U_n)) = 1] - \Pr[D(U_\ell) = 1]| < \frac{1}{p(n)} \quad (3.1)$$

A Course in Cryptography

3.3.1 Definition of a Pseudo-random Generators

► **Definition 77.1** (Pseudo-random Generator). A function $G : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a *Pseudo-random Generator (PRG)* if the following holds.

1. (efficiency): G can be computed in p.p.t.
2. (expansion): $|G(x)| > |x|$
3. The ensemble $\{x \leftarrow U_n : G(x)\}_n$ is pseudo-random.

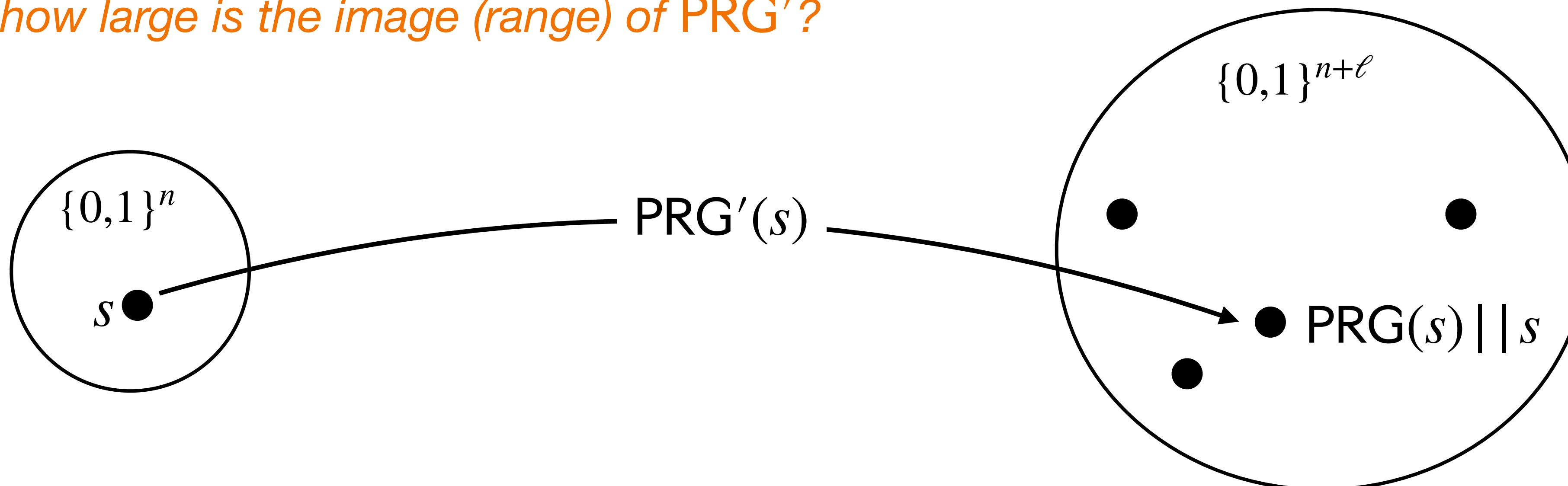
$$\text{Adv}(\mathcal{A}) = |\Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}| < \text{negl}(n)$$

Note on PRGs: Pseudorandom Strings Are Far From Random

Example

Consider the following candidate for a pseudorandom function defined as $\text{PRG}'(s) = \text{PRG}(s) \parallel s$.

🤔 *how large is the image (range) of PRG' ?*



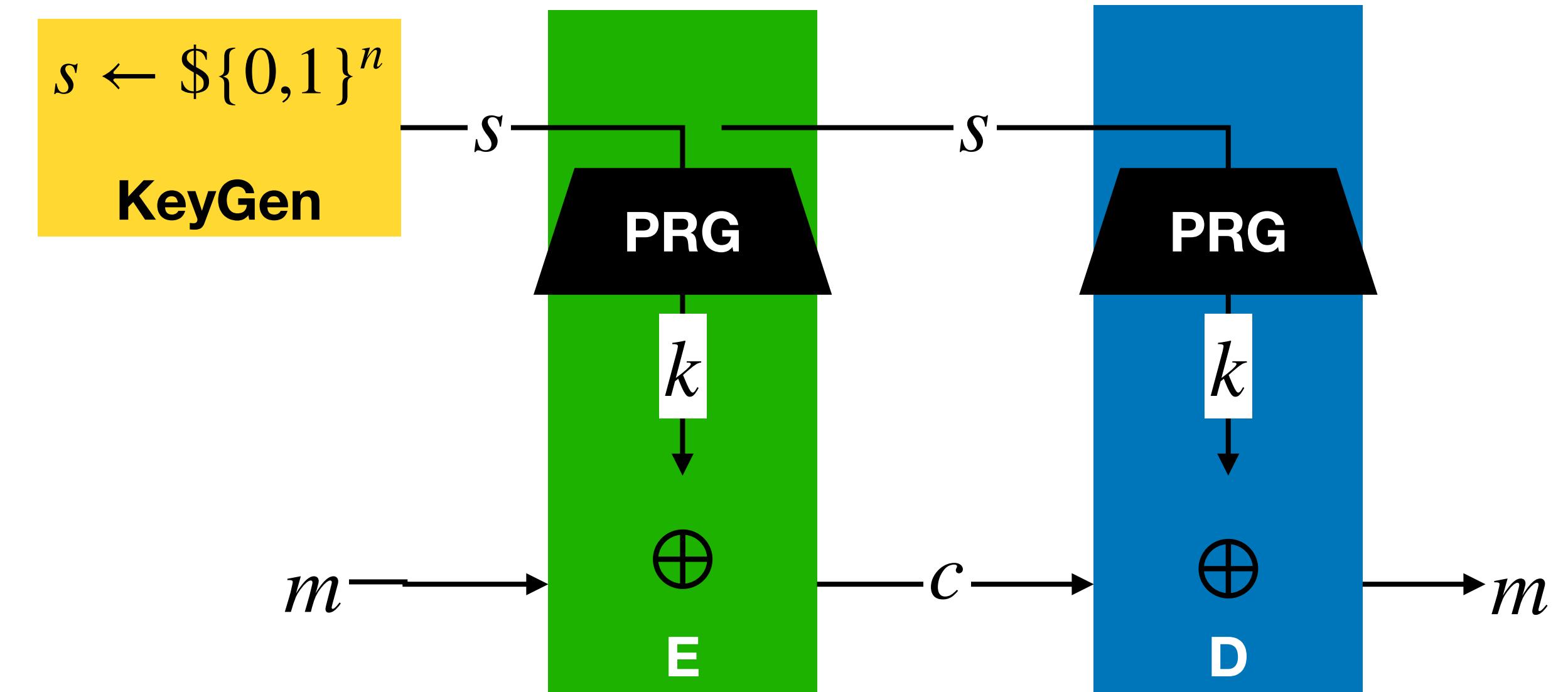
PRG' outputs strings of $n + \ell$ bits, but actually the size of its image is as large as its input so no more than 2^n strings can be generated by PRG' .

Constructing a Secure Encryption Scheme From a PRG

A generic PRG

$\text{PRG} : \{0,1\}^n \rightarrow \{0,1\}^\ell$

$\text{PRG}(s) = k$



A One-time PRG cipher

$\mathcal{M} = C = \{0,1\}^\ell, \mathcal{K} = \{0,1\}^n, n < \ell$

$\text{KeyGen}(1^n) \rightarrow s$ (where $s \leftarrow \$\{0,1\}^n$)

$\text{Enc}(s, m) = \text{PRG}(s) \oplus m$

$\text{Dec}(s, c) = \text{PRG}(s) \oplus c$

🧐 Is this cipher perfectly secure?

No! We need a **new security definition** that works when $|\mathcal{K}| < |\mathcal{M}|$

“Left OR Right” Security (Intuition)

WHITE

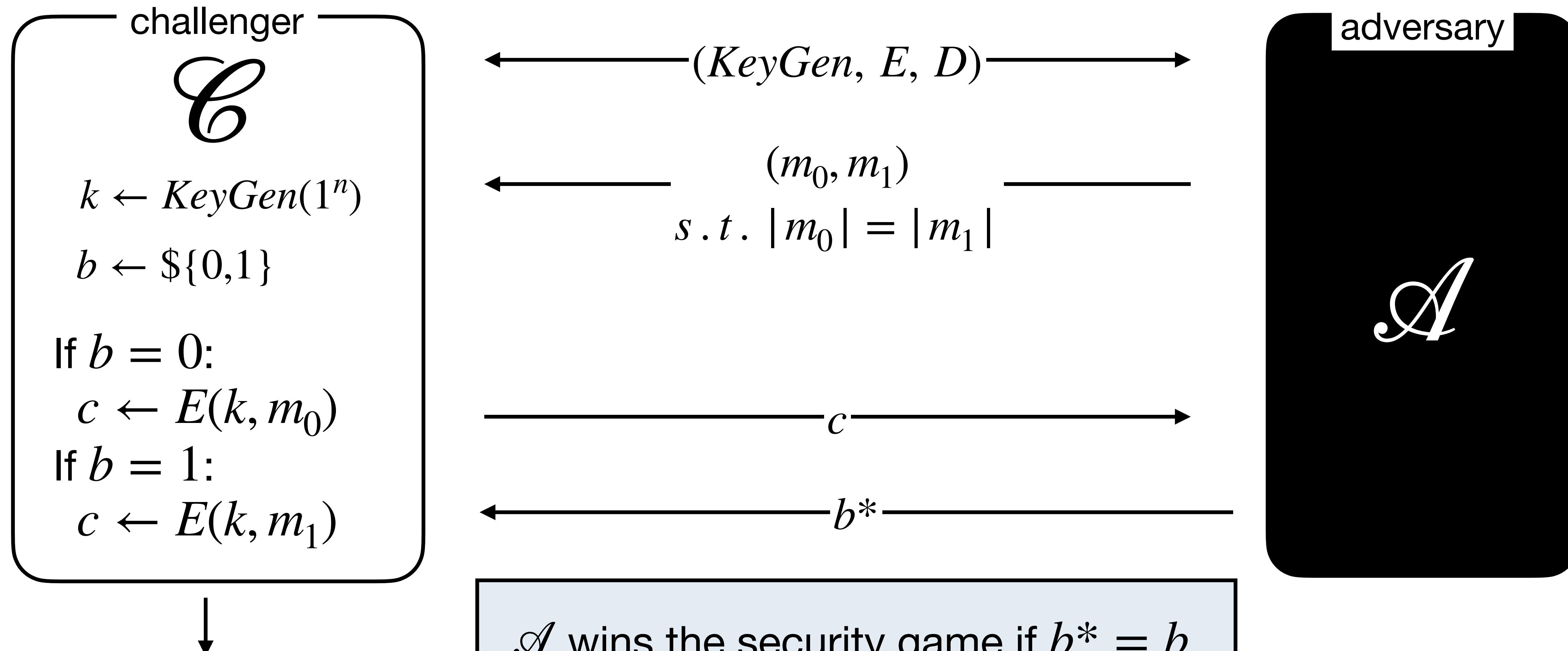


MAGENTA



Semantic Security (Aka Single-Message Secure Encryption)

Aim: quantify the attacker's likelihood in distinguishing an encryption of a (chosen) message m_0 from an encryption of another (chosen) message m_1



\mathcal{A} wins the security game if $b^* = b$.
If $b^* \neq b$, \mathcal{A} loses the game

Semantic Security for Symmetric Encryption

Definition: Semantic Security

A symmetric encryption scheme is semantically secure if any efficient attacker \mathcal{A} has only negligible advantage in winning the semantic security game. Formally,

$$Adv(\mathcal{A}) = \left| Pr[\mathcal{A} \text{ wins}] - \frac{1}{2} \right| < negl(n)$$

Verbose description of the semantic security game

1. The challenger \mathcal{C} generates a key $k \leftarrow KeyGen(1^n)$ and draws a random bit $b \leftarrow \$\{0,1\}$.
2. The adversary \mathcal{A} chooses two messages m_0, m_1 of the same length and sends them to \mathcal{C} .
3. \mathcal{C} encrypts m_b according to the bit drawn in step 1, and returns $c = Enc(k, m_b)$ to \mathcal{A} .
4. \mathcal{A} tries to determine b from c, m_0 , and m_1 .
5. \mathcal{A} sends b^* to the \mathcal{C} . The adversary wins if $b^* = b$.

Remarks on the Definition

Definition: Semantic security

A symmetric encryption scheme is semantically secure if any efficient attacker \mathcal{A} has only negligible advantage in winning the semantic security game. Formally,

$$Adv(\mathcal{A}) = \left| Pr[\mathcal{A} \text{ wins}] - \frac{1}{2} \right| < negl(n)$$

- We don't expect the encryption scheme to hide the length of the plaintext; (hence m_0 and m_1 must have the same length).
- An attacker who just guesses, choosing a random $b^* \leftarrow \$\{0,1\}$, has advantage 0.
- An attacker who always answers $b^* = 1$ (or $b^* = 0$) also has advantage 0.
- If the encryption scheme is the one time pad, any attacker has advantage 0.

Proving our Construction Is Semantically Secure

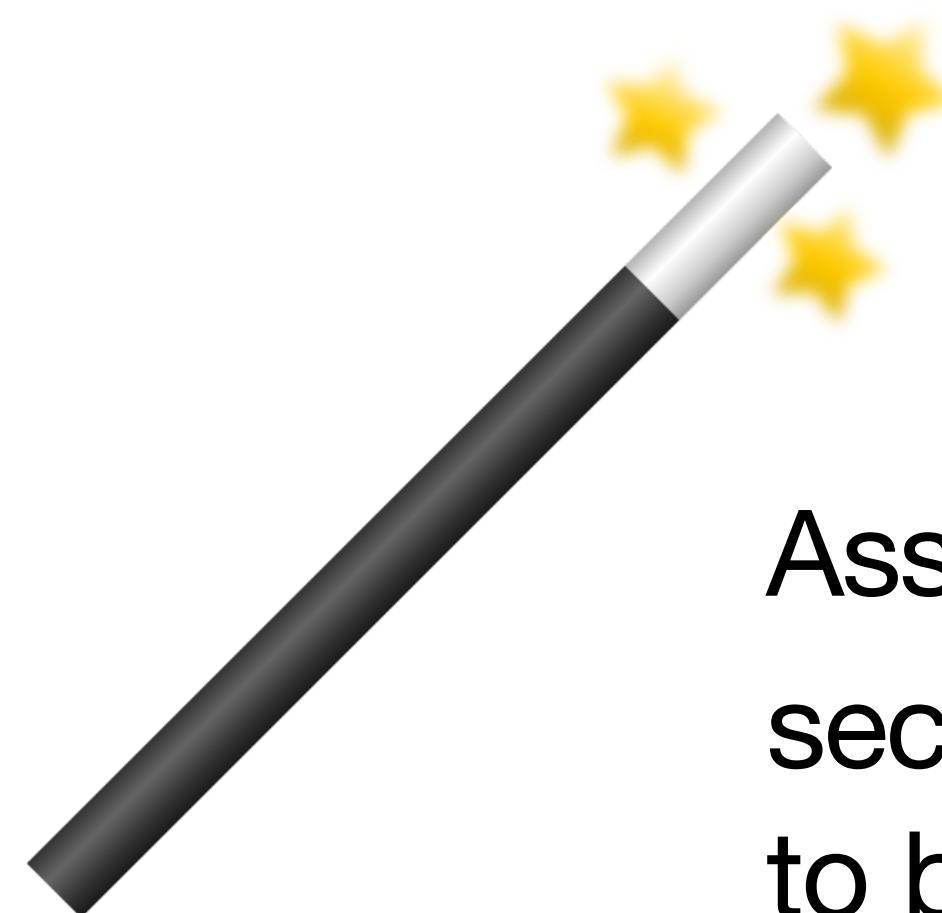
Theorem (secure encryption scheme with short keys)

If $\text{PRG} : \{0,1\}^n \rightarrow \{0,1\}^\ell$ is a secure PRG, then the cipher defined by

$\text{Enc}(s, m) = \text{PRG}(s) \oplus m; \quad \text{Dec}(s, c) = \text{PRG}(s) \oplus c$ is semantically secure.

Proof Plan:

We must prove that **any** efficient adversary against the encryption's semantical security has negligible advantage, **without** knowing anything about the adversary's strategy.



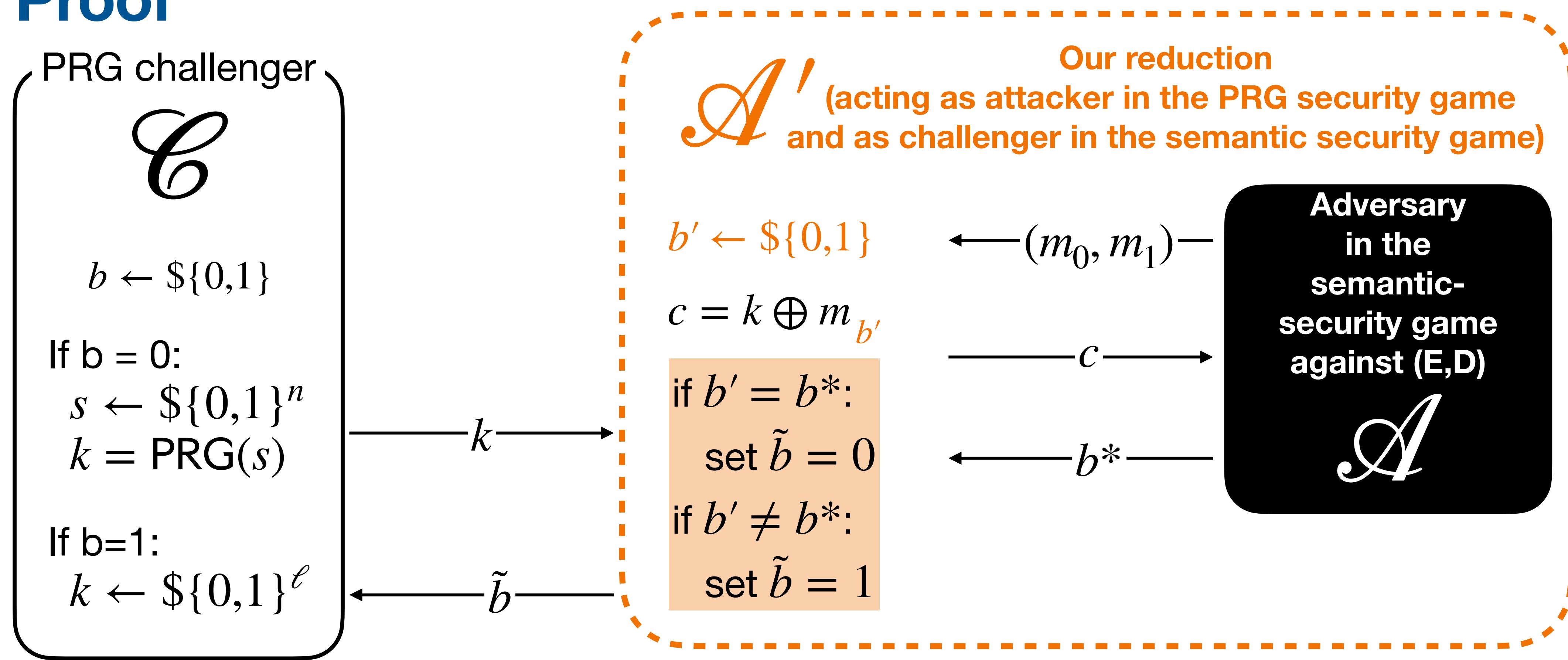
...or... *proof by reduction to absurd*

Assume that there exists an adversary \mathcal{A} that can break the semantic security of the encryption. Then we build a new adversary \mathcal{A}' that uses \mathcal{A} to break the security of the PRG. Since PRG is assumed to be secure, such \mathcal{A}' cannot exist. Thus it was absurd to assume \mathcal{A} exists in the first place.

The Proof

This proof [next 3 slides] is not part of the exam, but it is here to show you how “formal reduction proofs” look like. If you understand the mechanism and like the reasoning, then you have a good chance to become a cryptographer!

The Proof



Important observations

If $b = 0$, the ciphertext c is the encryption using the PRG cipher. Because we assumed that \mathcal{A} wins this game with non negligible probability this means $b' = b^*$. So \mathcal{A}' wins when \mathcal{A} does.

If $b = 1$, \mathcal{A}' encryption is the **OTP** (perfectly secure), thus \mathcal{A} has no advantage. So \mathcal{A}' only guesses correctly with $1/2$ probability (0 advantage).

The Proof

$$\Pr[\mathcal{A}' \text{ wins}] = \Pr[\mathcal{A}' \text{ wins AND } b = 0] + \Pr[\mathcal{A}' \text{ wins AND } b = 1]$$

conditional probability

$$Pr[A | B] = \frac{Pr[A \cap B]}{Pr[B]}$$

$$= \Pr[\mathcal{A}' \text{ wins } | b = 0] \Pr[b=0] + \Pr[\mathcal{A}' \text{ wins } | b = 1] \Pr[b=1]$$

PRG-cipher

1/2

*OTP-cipher
(perfect security)*

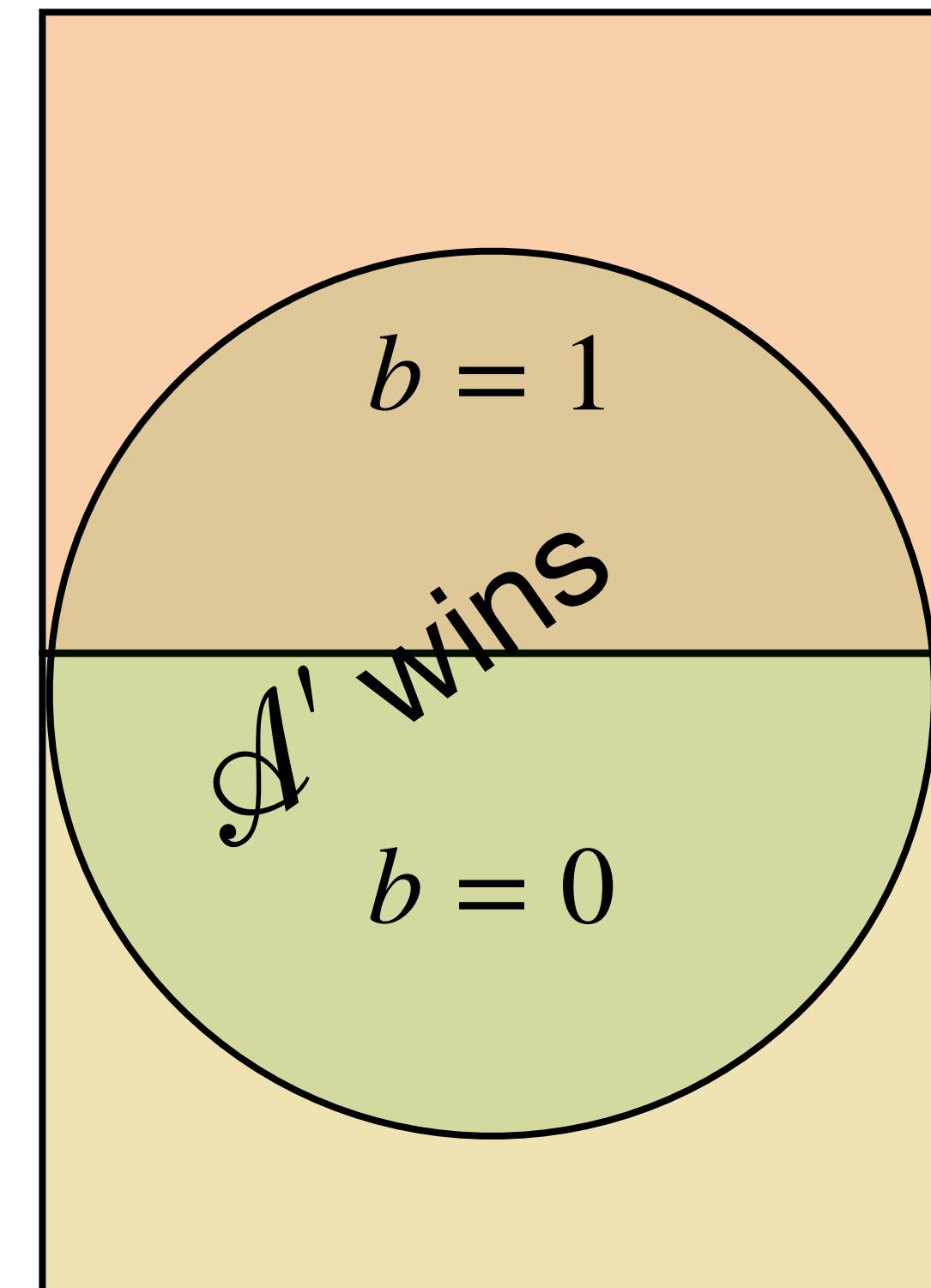
1/2

$\Pr[\mathcal{A} \text{ wins sem.sec game against } (\mathbf{E}, \mathbf{D})]$

Thus $\Pr[\mathcal{A}' \text{ wins PRG}] = \Pr[\mathcal{A} \text{ wins sem.sec}] \cdot (1/2) + 1/2 \cdot (1/2)$

Or, reorganising the terms: $\Pr[\mathcal{A} \text{ wins sem.sec}] = 2 \Pr[\mathcal{A}' \text{ wins PRG}] - 1/2$

$$Adv_{sem.sec}(\mathcal{A}) = \left| \Pr[\mathcal{A} \text{ wins}] - \frac{1}{2} \right| = \left| (2\Pr[\mathcal{A}' \text{ wins PRG}] - 1/2) - \frac{1}{2} \right| = 2 \cdot Adv_{PRG}(\mathcal{A}')$$



The Proof

$$Adv_{sem.sec}(\mathcal{A}) = \left| Pr[\mathcal{A} \text{ wins}] - \frac{1}{2} \right| = \left| (2Pr[\mathcal{A}' \text{ wins PRG}] - 1/2) - \frac{1}{2} \right| = 2 \cdot Adv_{PRG}(\mathcal{A}')$$

This concludes the proof of the theorem 



WHY ?

If our PRG-based encryption is not secure, then \mathcal{A} has a non-negligible advantage in winning the semantic security game. If that was the case, we have constructed an efficient (PPT) reduction/adversary \mathcal{A}' that uses \mathcal{A} to win the PRG security game and succeed with half of the advantage of \mathcal{A} . Since we assumed the PRG to be secure, it is impossible for any efficient adversary to break the PRG. Therefore, such an \mathcal{A}' cannot exist. Which in turn implies that \mathcal{A} cannot exist. So it was absurd to assume such an \mathcal{A} existed in the first place.

This reasoning implies that our PRG-based encryption is *provably* secure.

We've got secure communication!



Are we done yet?

- Multi-Message Security
- Messages of different length
- Message integrity (Lecture 4)
- Shared secret key (Module 2)

Lecture 2 - Agenda

Pseudorandom Generators (PRG)

- Definition
- Security (Real or Random?)
- Negligible
- The RSA PRG
- Secure encryption from PRG
- Semantic Security (Left or Right)
- PRG-Based Encryption [Proof]

Block Ciphers

- Pseudo Random Permutations
- Definition of Block Cipher
- Chosen Plaintext Attack (IND-CPA)

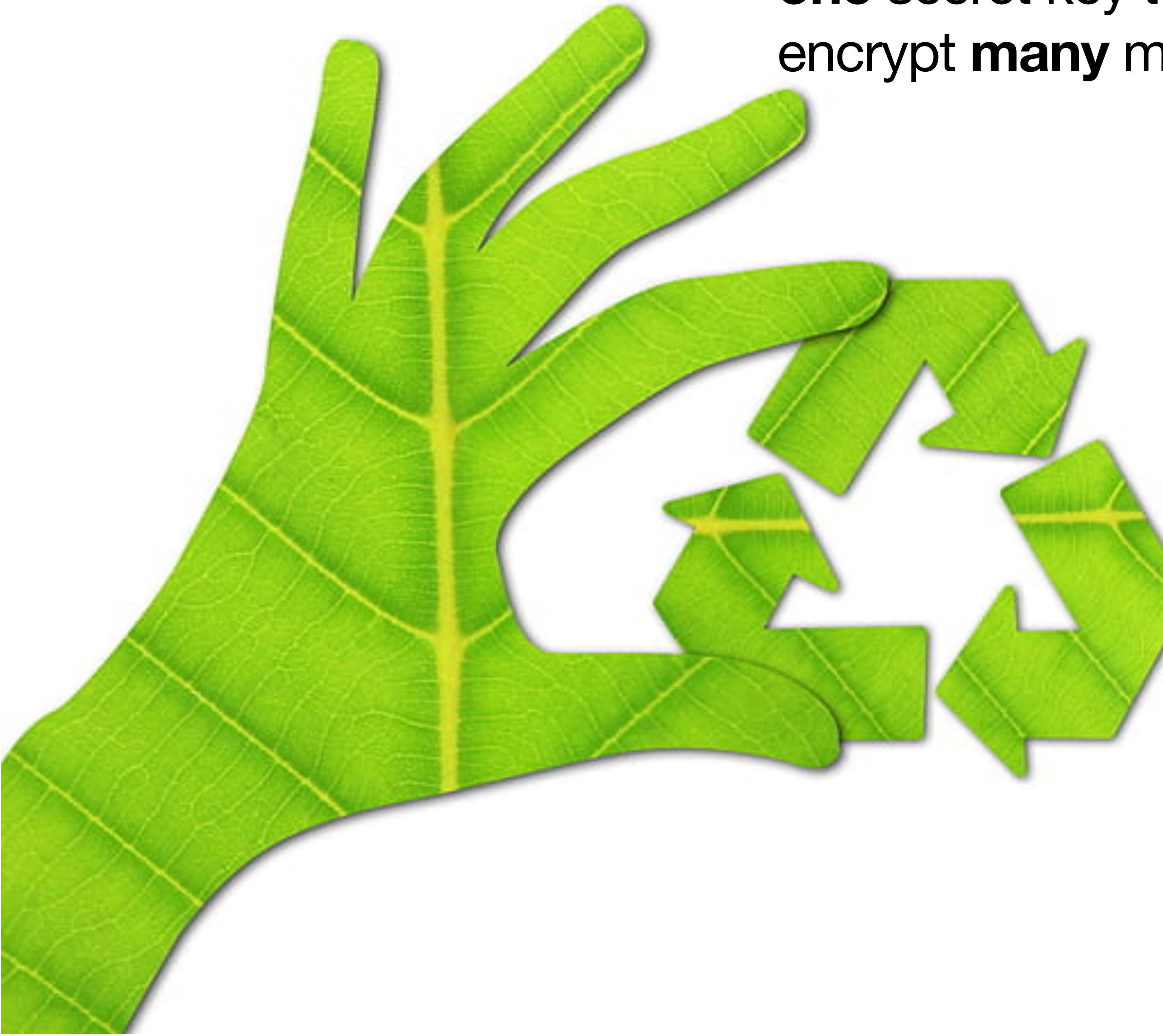
Block Ciphers (Continued)

- Advanced Encryption Standard
- Design Principles

Modes of Operation

- ECB, CBC, CTR
- Is AES-ECB Semantically Secure?

Efficient (and Secure) Encryption



one secret key to
encrypt **many** messages

one-time symmetric encryption
With shorter keys (from PRG)

REUSE
REDUCE
RECYCLE

Mechanisms to “evolve” one key

A Candidate for Multi-Message Secure Encryption

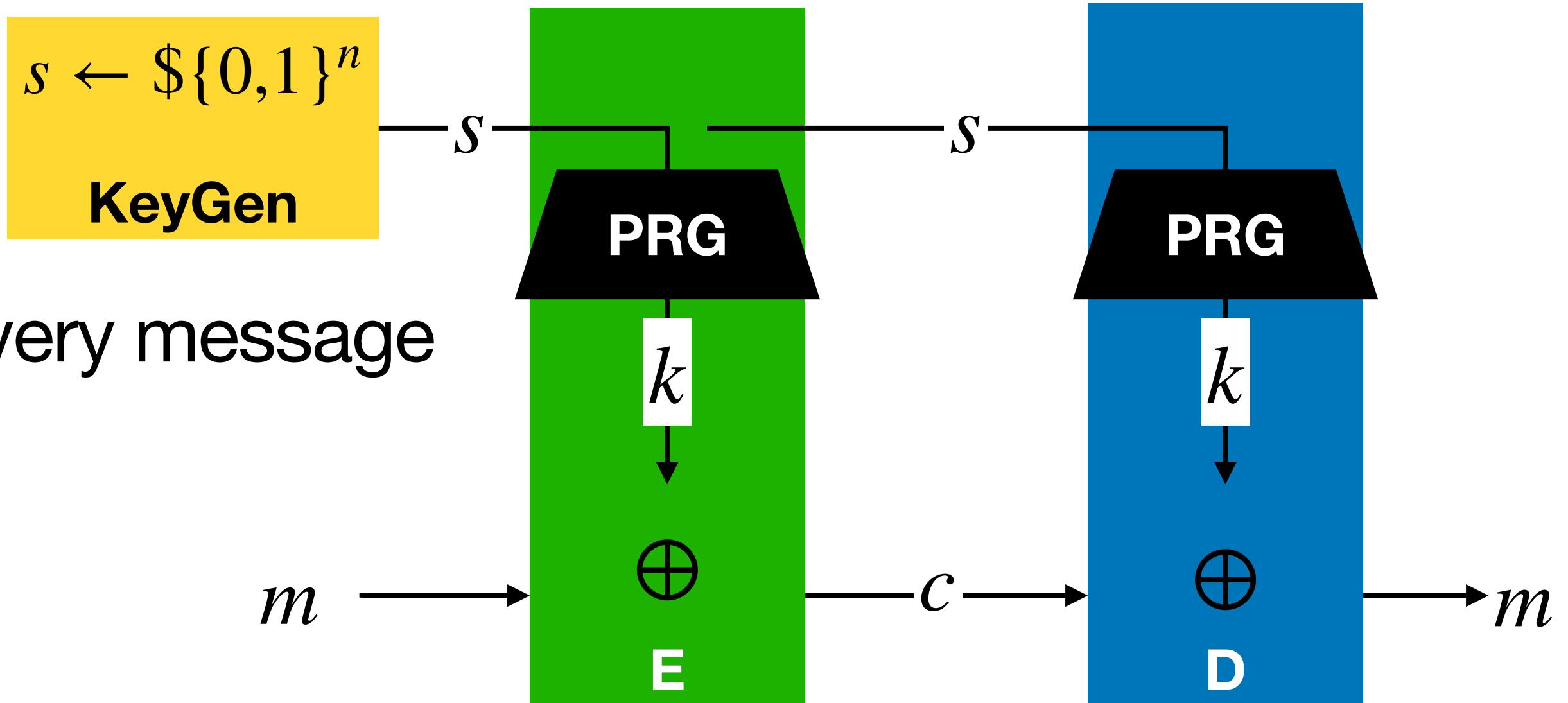
1. Define encryption as $E(s, m) = (s \parallel \text{PRG}(s) \oplus m) = c$

for a random $s \leftarrow \$\{0,1\}^n$, different for every message



There's a problem...

One-Message Secure encryption (OTP+PRG)



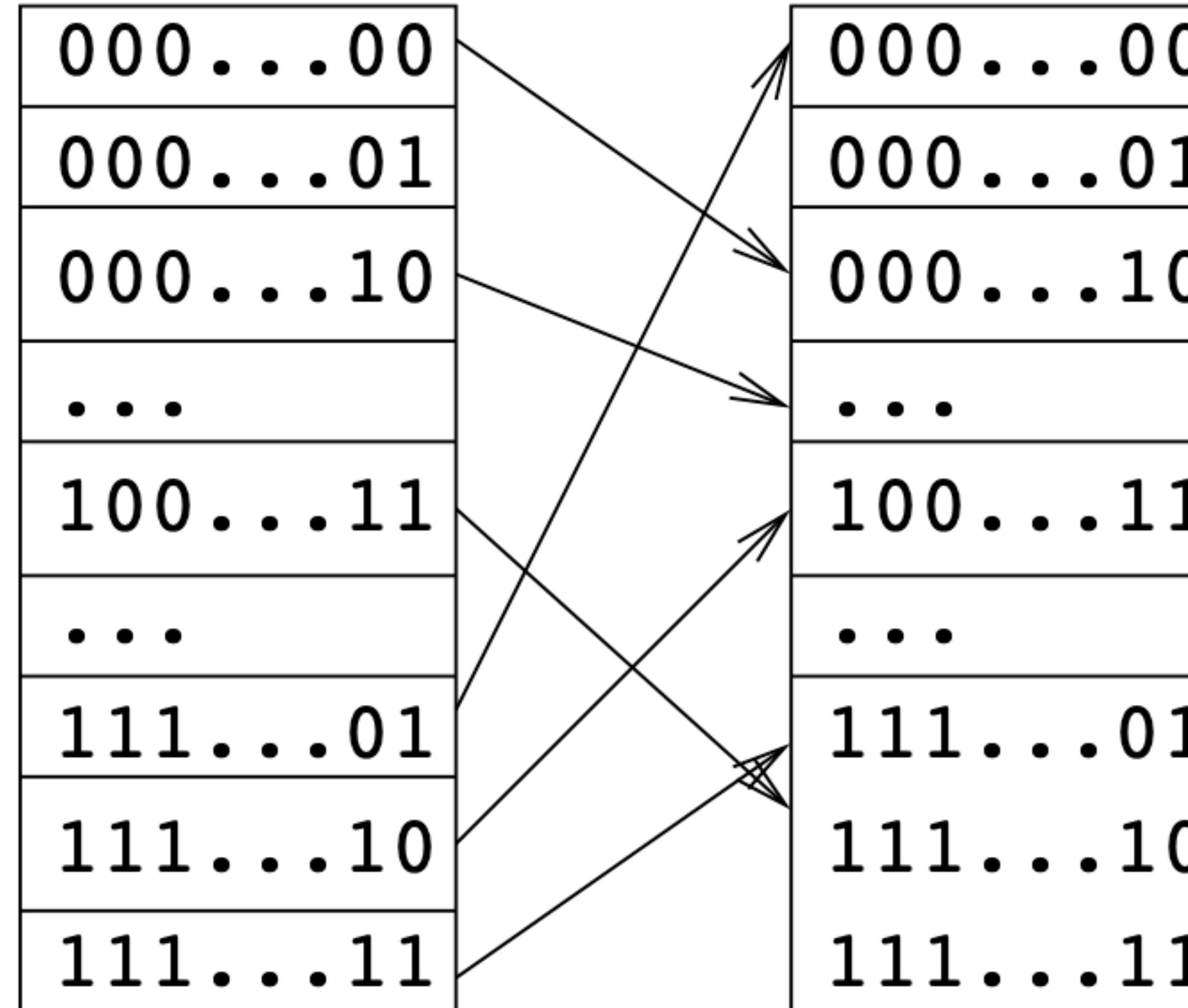
2. Find a way to make the PRG depend on a shared key.

This object is called Pseudo Random Permutation $f_k : \{0,1\}^n \rightarrow \{0,1\}^n$.

Encryption is defined as $c = (s \parallel f_k(s) \oplus m)$ for a random s (different for every message).

```
KeyGen(1K) : k ← $0,1K
E(k, m)   : Pick s ← $0,1n. Output c = (s, fk(s) ⊕ m)
D(k, c)   : Output c[1] ⊕ fk(c[0])
```

What Is a Pseudo Random Permutation (PRP)?



A **random permutation** is a permutation chosen uniformly at random from the set $\text{Perms}(\mathcal{M})$ of all permutations (one-to-one maps) on the set $\mathcal{M} = \{0,1\}^n$.

Informal:

A family of **pseudo-random permutations** $\{f_k : \{0,1\}^n \rightarrow \{0,1\}^n\}_{k \in \{0,1\}^K}$ is a collection of permutations such that:

- 1) each f_k can be *efficiently computed*; and
- 2) there is *no efficient way to distinguish* between f_k and a random function.

💡 How many possible permutations over $\mathcal{M} = \{0,1\}^n$ are there? $|\text{Perms}(\mathcal{M})|$?

$$|\text{Perms}(\mathcal{M})| = 2^n! \approx 2^{n2^n} \text{ vs } |\text{PRP}| = |\{0,1\}^K| = 2^K$$

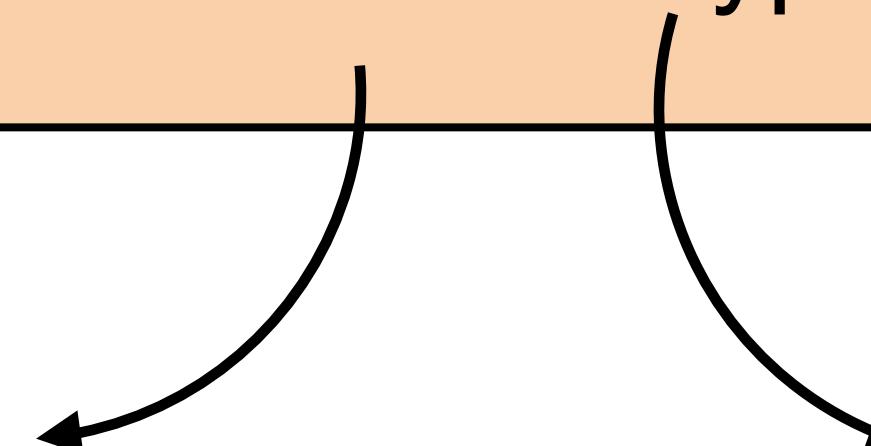
Pseudo Random Permutation

```
KeyGen( $1^k$ ) :  $k \leftarrow \{0,1\}^k$ 
E( $k, m$ ) : Pick  $s \leftarrow \{0,1\}^n$ . Output  $c = (s, f_k(s) \oplus m)$ 
D( $k, c$ ) : Output  $c[1] \oplus f_k(c[0])$ 
```

Theorem

If f_k is a PRP then the scheme (KeyGen, E, D) is a secure encryption scheme.

we need a *new* definition of security for multi messages



Let's *refine* the definition of an encryption scheme

Block Ciphers

Definition: Block Cipher

A Block Cipher is a **keyed** function that is **deterministic** and **invertible**.

Formally, $E : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{M}$, where $\mathcal{K} = \{0,1\}^K$ is a key space, $\mathcal{M} = \{0,1\}^n$ is the block space and for every $k \in \mathcal{K}$, the function $E(k, \cdot) : \{0,1\}^n \rightarrow \{0,1\}^n$ is invertible, that is to say, there exist an efficient function $D(k, \cdot) : \{0,1\}^n \rightarrow \{0,1\}^n$ such that $D(k, E(k, m)) = m$ for every $m \in \{0,1\}^n$.

Plaintexts and ciphertexts are both called **blocks**.



What if the plaintext message is longer than one block (n -bits) ?

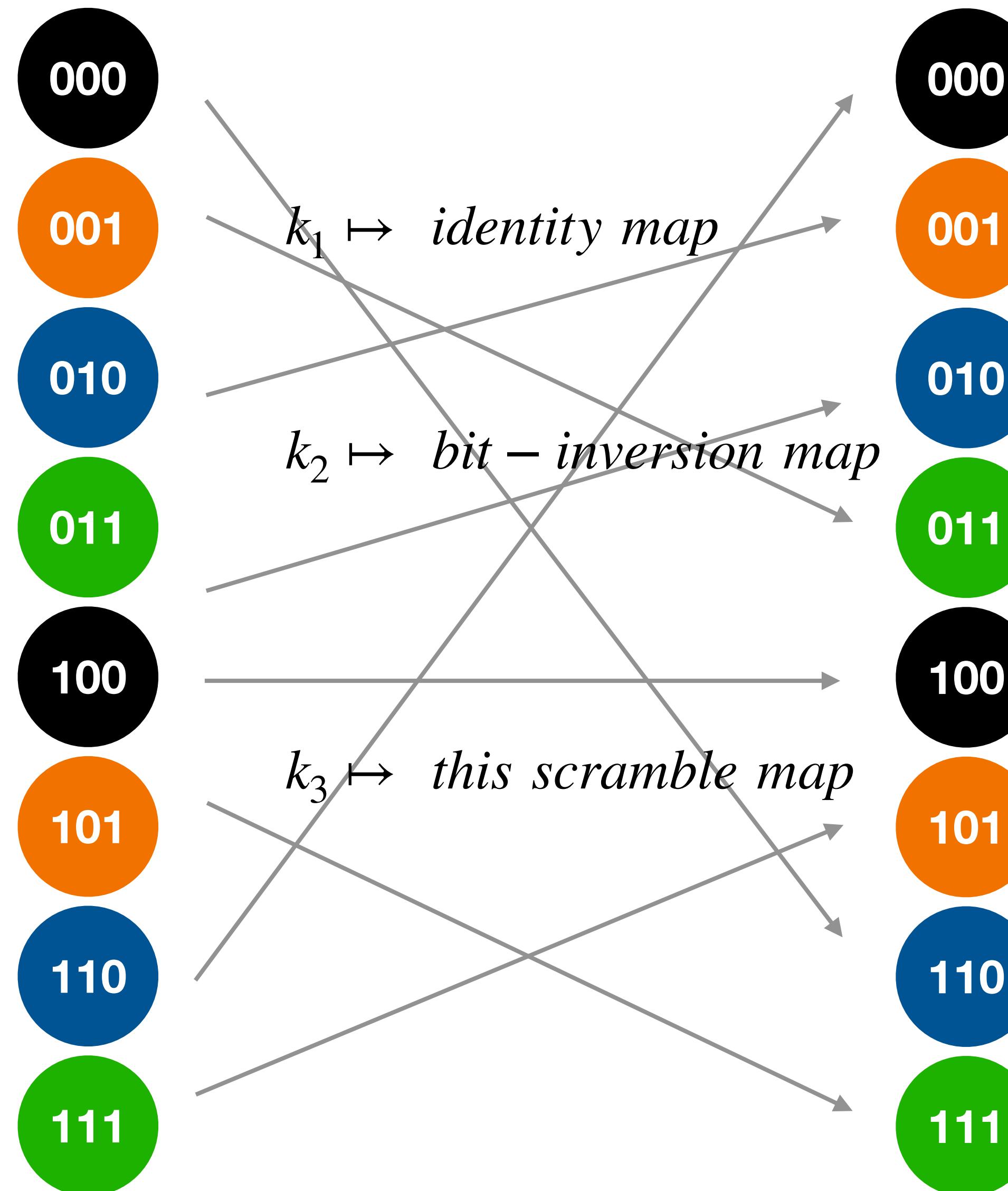
The Block cipher “chains” blocks according to a “mode of operation” (more on this later)

Observation: we can now reuse the same key to encrypt/decrypt multiple messages!

Block Ciphers - Examples

$$\{0,1\}^{n=3} \dashrightarrow E(k, \cdot) \dashrightarrow \{0,1\}^{n=3}$$

🧐 How many possible keys (encryption maps) are there?



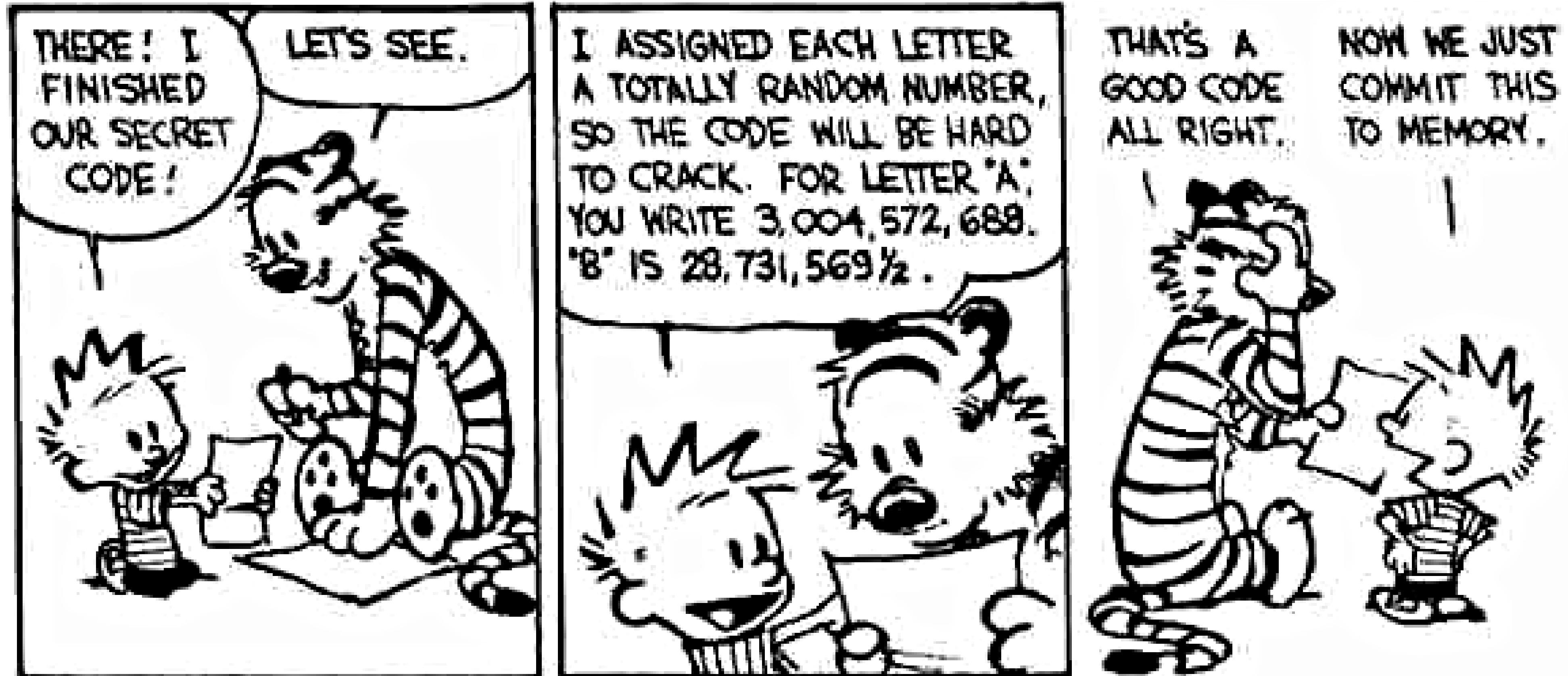
The total number of invertible functions over $\{0,1\}^3$ is
 $8! = 8 \cdot 7 \cdots \cdot 2 \cdot 1 = 40\,320$

Large enough?!



Bad news: $8! \sim 2^{15}$ is a manageable for modern computers
Typical values today are $|\mathcal{K}| = 2^{128}$ or 2^{256}

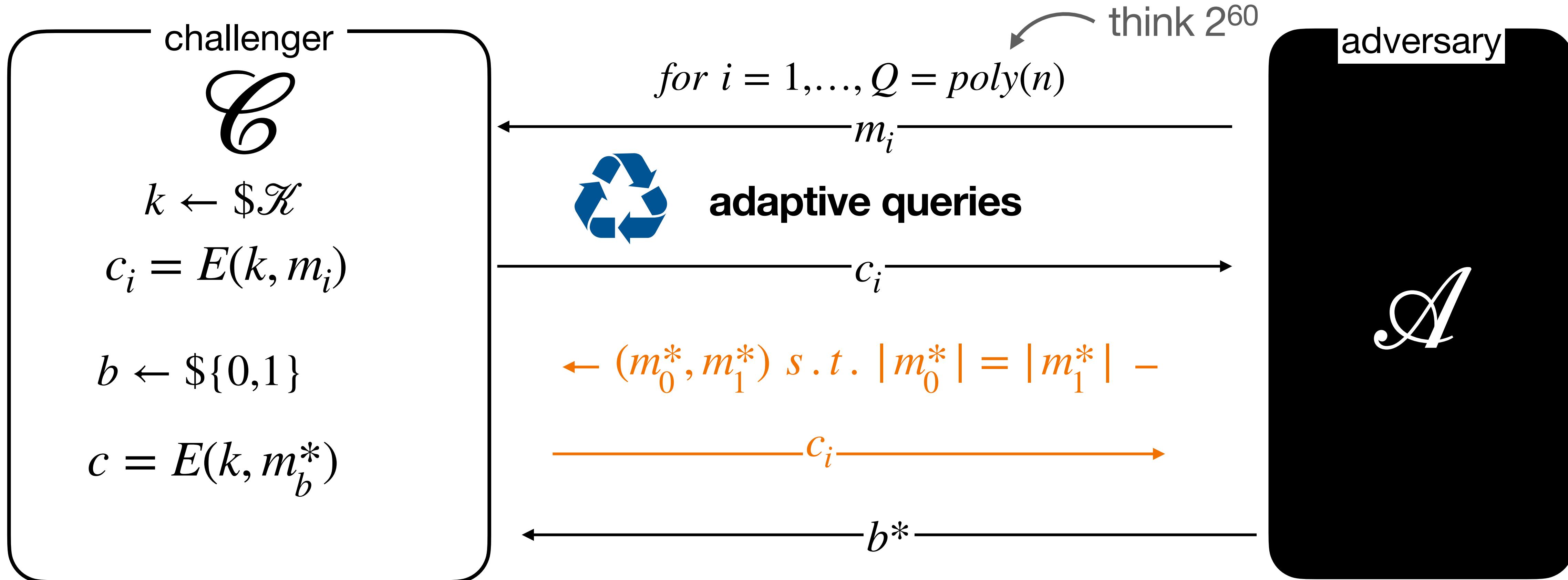
Security vs Efficiency - Finding the Right Balance



We cannot (efficiently) implement (all possible) random permutations for reasonable sizes of n . Instead, we strive to construct ciphers that **cannot be distinguished from random permutations**.

Chosen Plaintext Attack (Security Notion for Block Ciphers)

Aim: quantify the \mathcal{A} 's likelihood in distinguishing between encryptions of two messages, given that \mathcal{A} can see encryptions of any messages of its choice.



\mathcal{A} wins the security game iff $b^* = b$. If $b^* \neq b$, \mathcal{A} loses the game

Lecture 2 - Agenda

Pseudorandom Generators (PRG)

- Definition
- Security (Real or Random?)
- Negligible
- The RSA PRG
- Secure encryption from PRG
- Semantic Security (Left or Right)
- PRG-Based Encryption [Proof]

Block Ciphers

- Pseudo Random Permutations
- Definition of Block Cipher
- Chosen Plaintext Attack (IND-CPA)

Block Ciphers (Continued)

- Advanced Encryption Standard
- Design Principles

Modes of Operation

- ECB, CBC, CTR
- Is AES-ECB Semantically Secure?

Examples of Block Ciphers

Data Encryption Standard (DES): is the most well-known symmetric-key block cipher. Introduced in the mid 1970s (refinement of IBM's Lucifer cipher) as the first commercial-grade modern algorithm with openly and fully specified implementation details. It is defined by the American standard FIPS 46-2.

NOW DEPRECATED DUE TO SHORT KEYS!!

Advanced Encryption Standard (AES) - aka Rijndael

Other modern secure symmetric ciphers, used more rarely than AES:

Serpent, Twofish, Camellia, RC5...

The Block Cipher *Par Excellence*: AES

Advanced Encryption Standard

```
function AESK(M)
(K_0, ..., K_10) ← expand(K)
S ← M ⊕ K_0

for r = 1 to 10 do
    S ← substitute-nibbles(S)
    S ← shift-rows(S)
    if (r≤9) then
        S ← mix-cols(S)
    fi
    S ← S ⊕ K_r
endfor

return S
```

key of reduced size

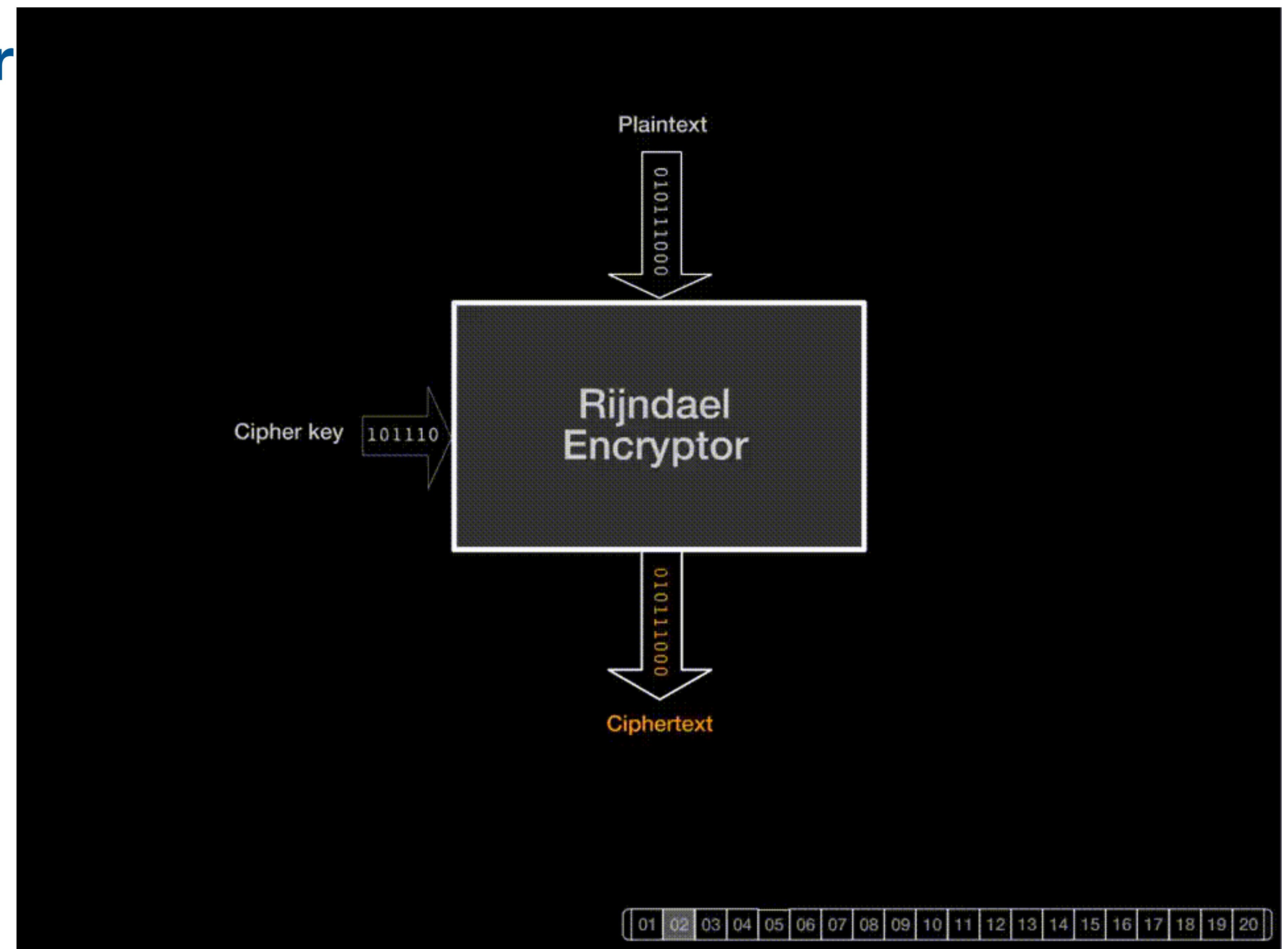
reuse material

OTP-style recycle

AES Block Cipher (Rijndael Design)

A bit of history:

- 1997 NIST opens a call for a new block cipher standard on 128-bit blocks
- 15 submissions
- 2 rounds of peer-review
- 5 finalists by 1999
- Intense cryptanalysis
- 2000 winner Rijndael
- AES official standard Nov. 2001 (FIPS197)



01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

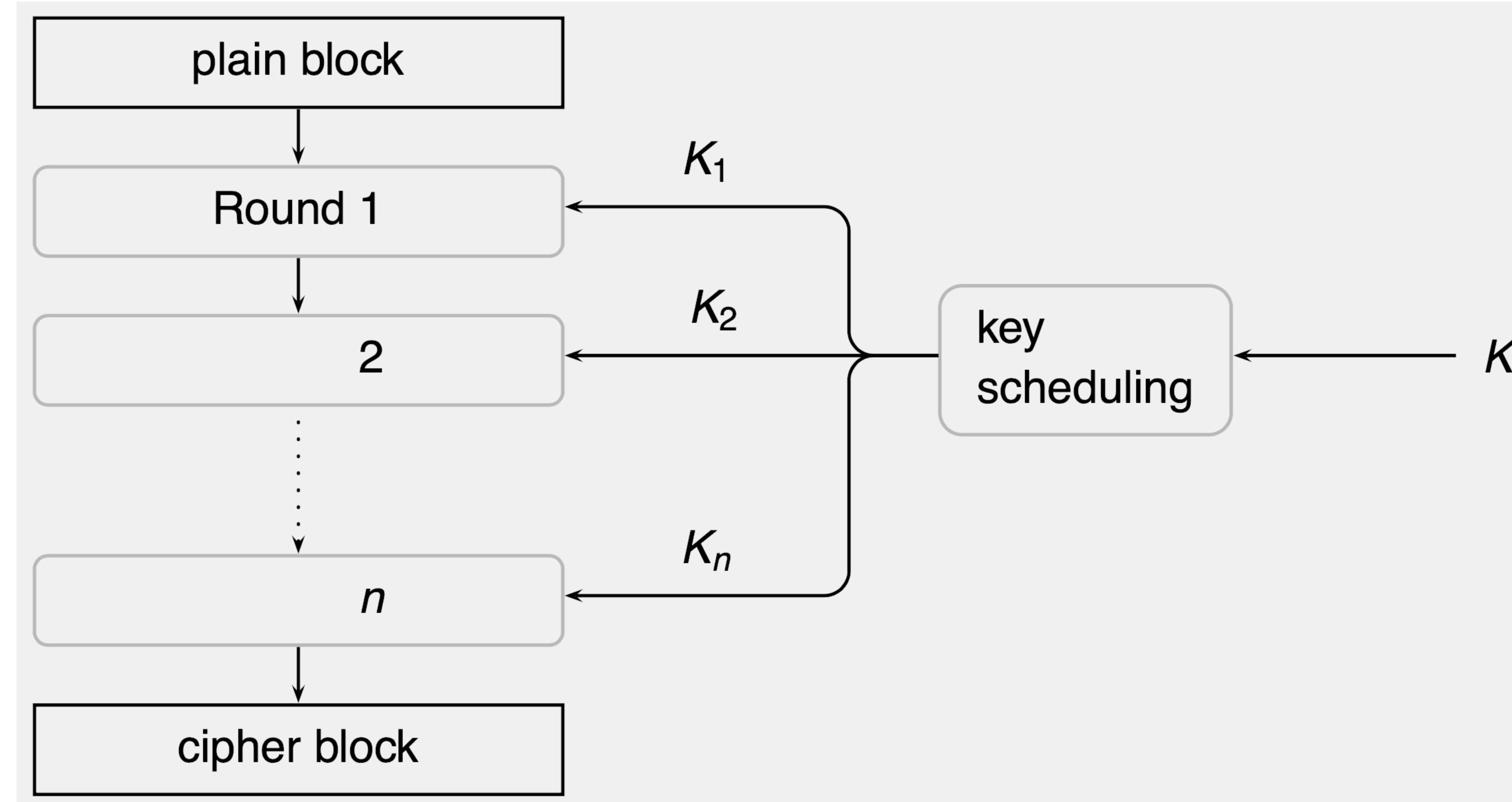
Why all of This?



If you cannot be a *truly* random permutation, at least strive to look like one.

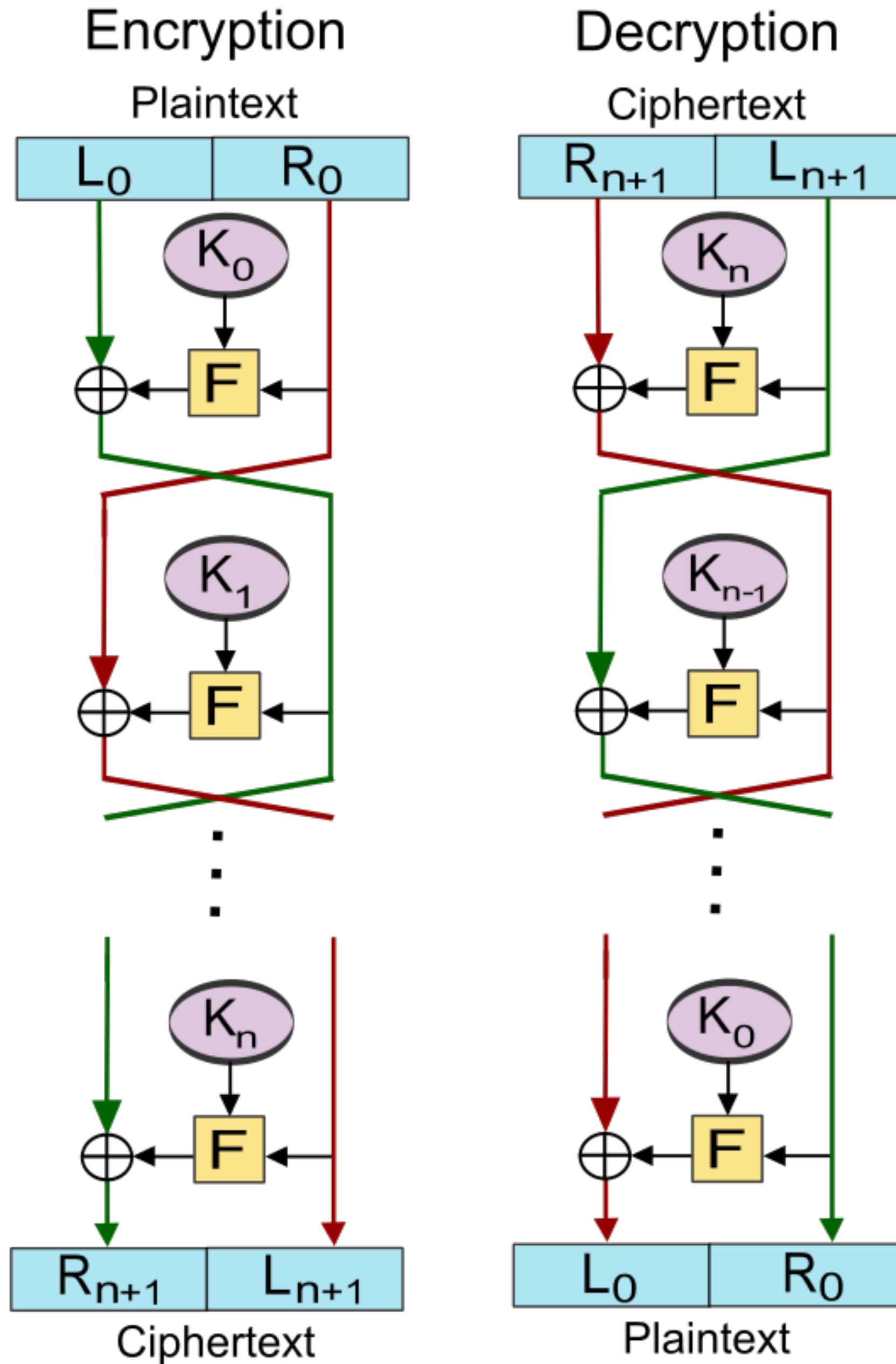
Design Principles for Block Ciphers: Iteration

Iteration: repeatedly apply a not-so-strong block cipher, each time with a different key (rounds)



This technique is not *provably* secure, but *heuristics* show that it works!

Design Principles for Block Ciphers: Feistel Networks



Feistel Network: The cipher function **F** is the same for every round.

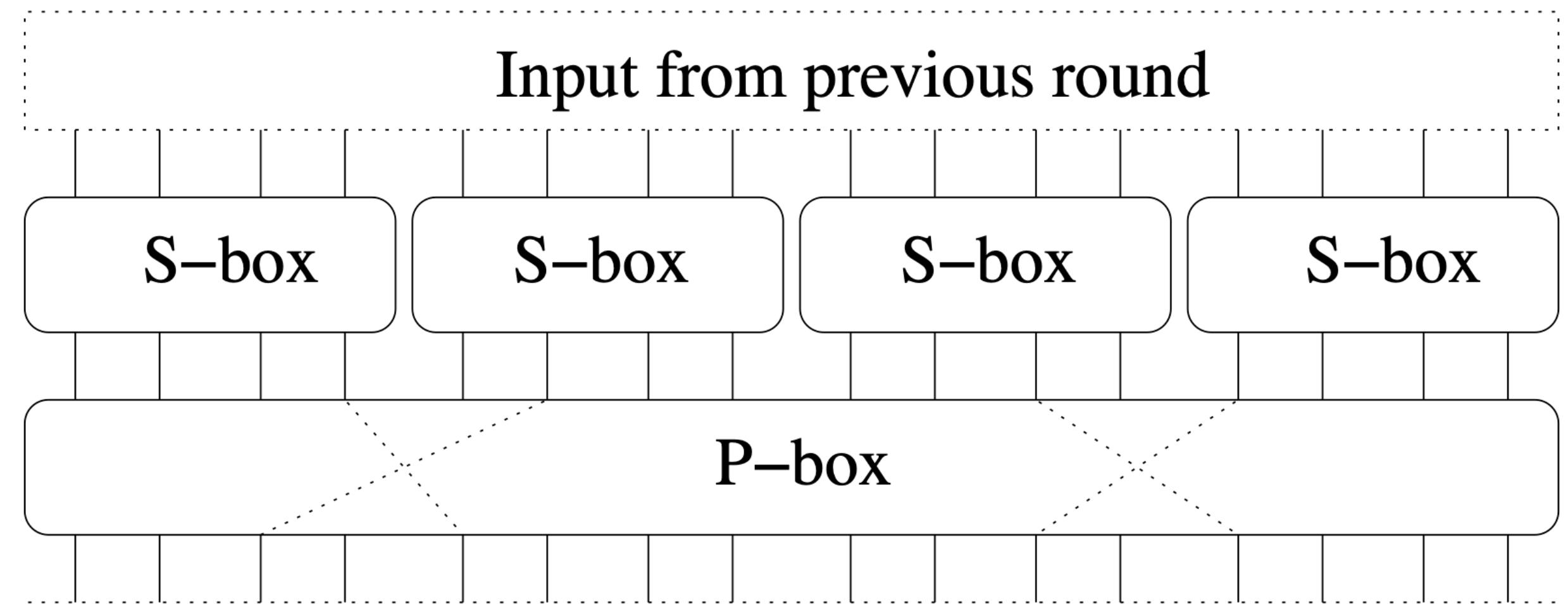
- **F** does not need to be invertible for the round to be invertible.
- **Decryption = Encryption** with round-keys in *reverse order*.

$$\begin{cases} L_{i+1} = R_i \\ R_{i+1} = L_i \oplus F(K_i, R_i) \end{cases}$$

Design Principles for Block Ciphers: Confusion & Diffusion

Confusion (S-boxes): Divide the n bits of input into b sub-blocks of n/b bits. Within each sub-block, apply a substitution table (**S-box**), i.e., a permutation on $\{0,1\}^{n/b}$, typically implemented as a lookup table. This introduces **confusion** to the cipher.

Diffusion (P-boxes): Confusion is local; to spread its effect apply a transposition **P-box**, permuting bits between sub-blocks. This introduces **diffusion**.



Lecture 2 - Agenda

Pseudorandom Generators (PRG)

- Definition
- Security (Real or Random?)
- Negligible
- The RSA PRG
- Secure encryption from PRG
- Semantic Security (Left or Right)
- PRG-Based Encryption [Proof]

Block Ciphers

- Pseudo Random Permutations
- Definition of Block Cipher
- Chosen Plaintext Attack (IND-CPA)

Block Ciphers (Continued)

- Advanced Encryption Standard
- Design Principles

Modes of Operation

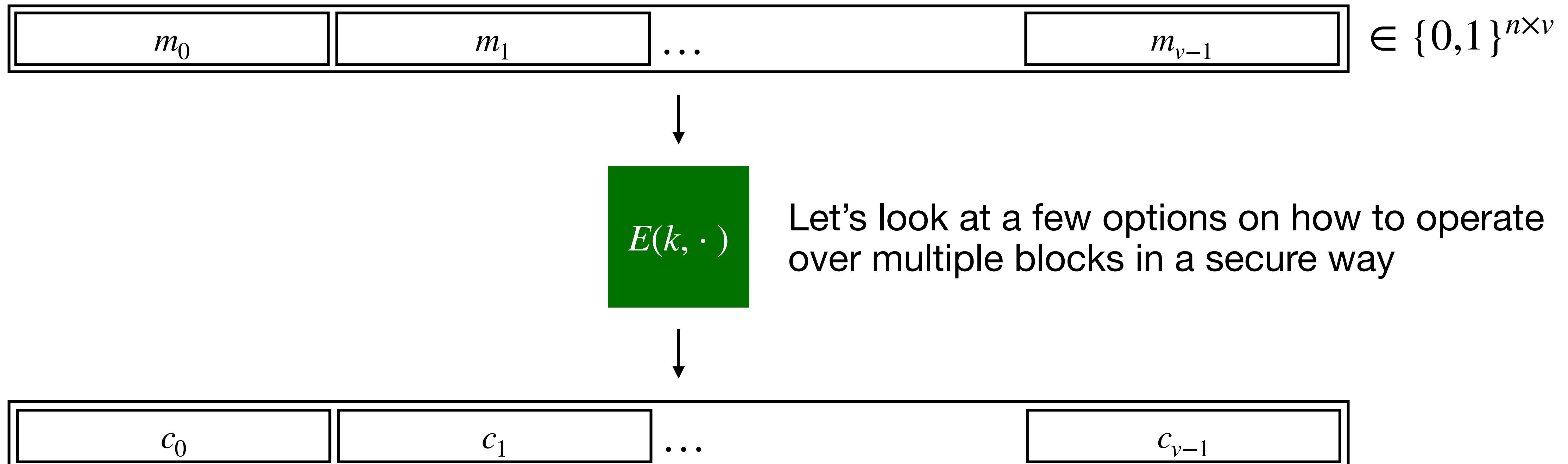
- ECB, CBC, CTR
- Is AES-ECB Semantically Secure?

Encrypting Long Messages (or Multiple Messages)

With the Same Key, Securely

A Block Cipher is a **deterministic, keyed** function that is **invertible**.

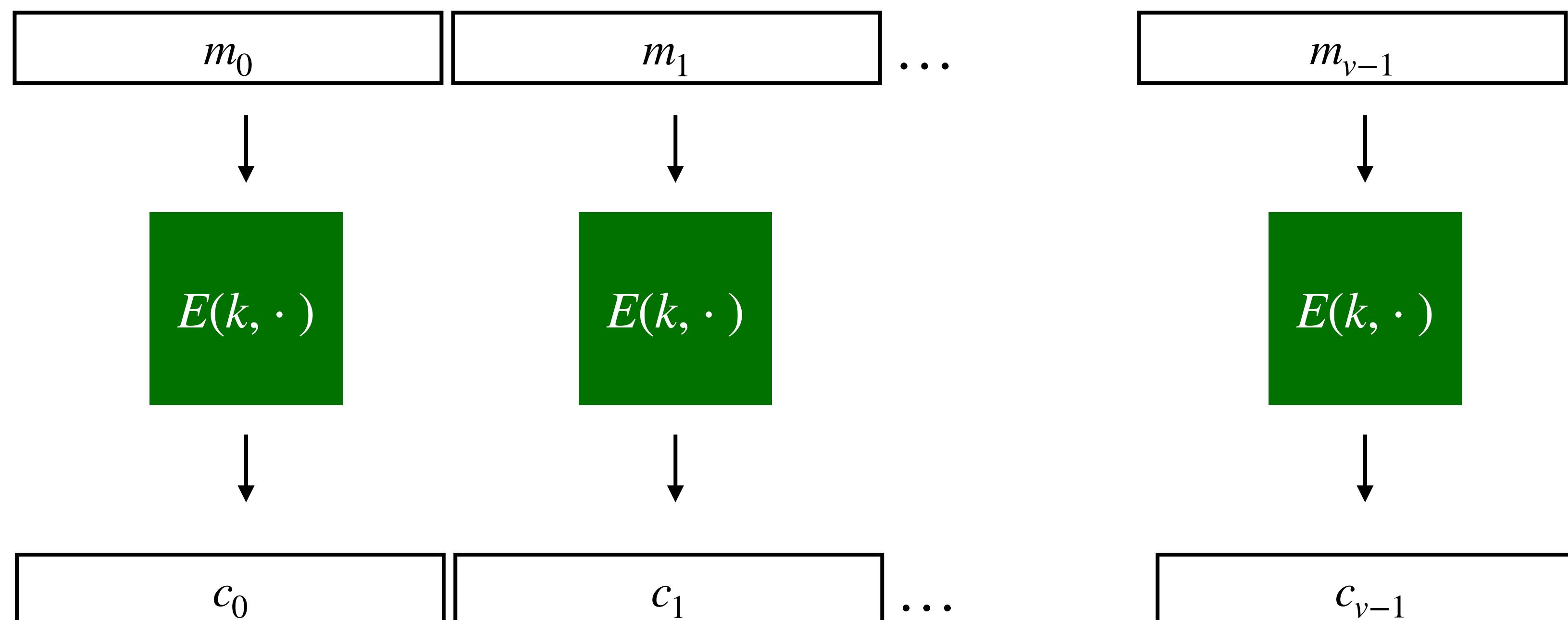
$$E(k, \cdot) : \{0,1\}^n \rightarrow \{0,1\}^n$$



Electronic Code Book Mode (ECB)

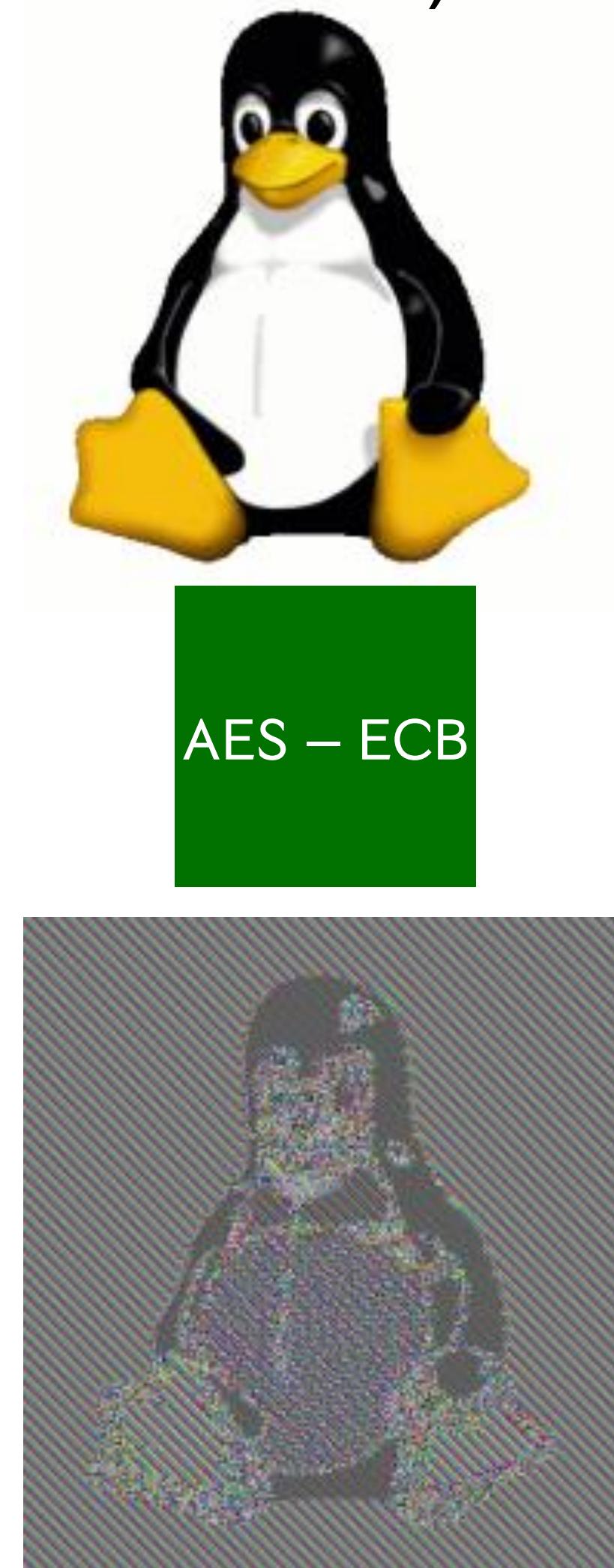
- + very easy to understand and implement
- + encryption and decryption are parallelizable (important for large data)
- lacks diffusion (it encrypts identical plaintext blocks into identical ciphertext blocks)

ECB is not recommended for use in cryptographic protocols.



💡 How can we express this with a formula?

$$c_i = E(k, m_i), \quad m_i = D(k, c_i), \quad \text{for } i = 0, \dots, v - 1$$



Cipher Block Chaining Mode (CBC)

⚠ formula?

- + Decryption is parallelizable
- Encryption is sequential (not parallelizable)

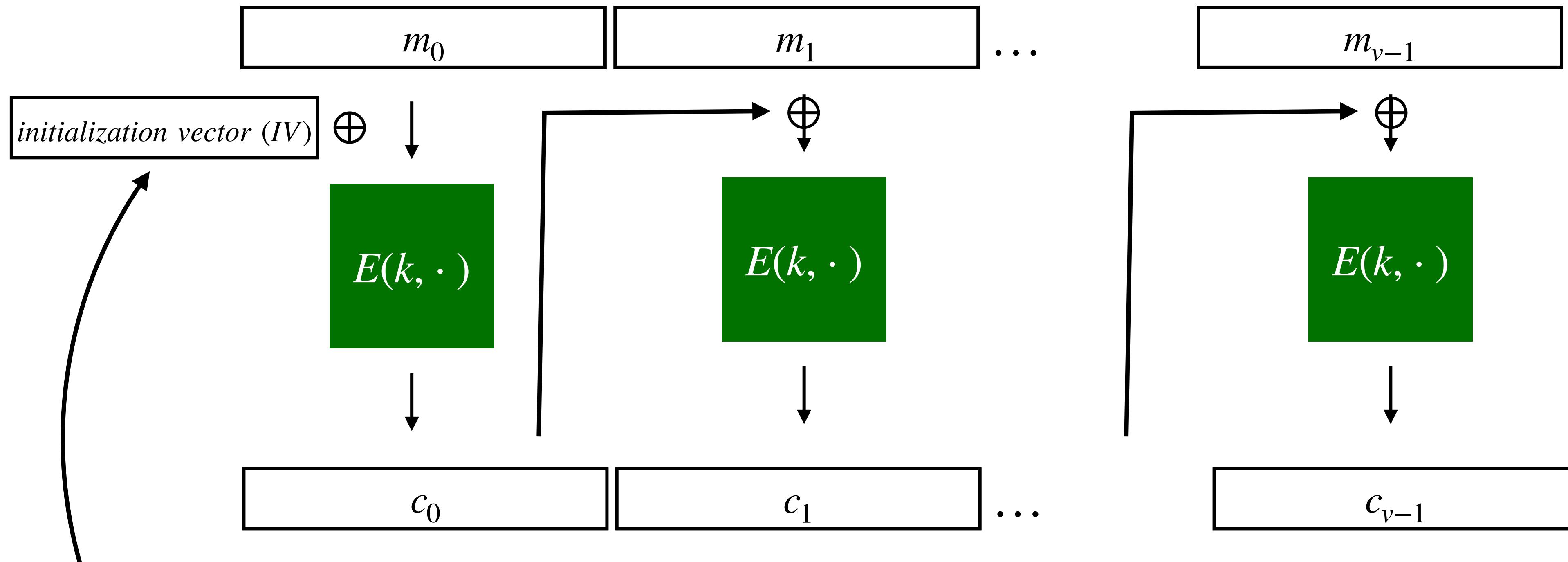
CBC is OK to use, but there's better.

$$c_0 = E(k, m_0 \oplus IV)$$

$$c_i = E(k, m_i \oplus c_{i-1}) \text{ for } i > 0$$

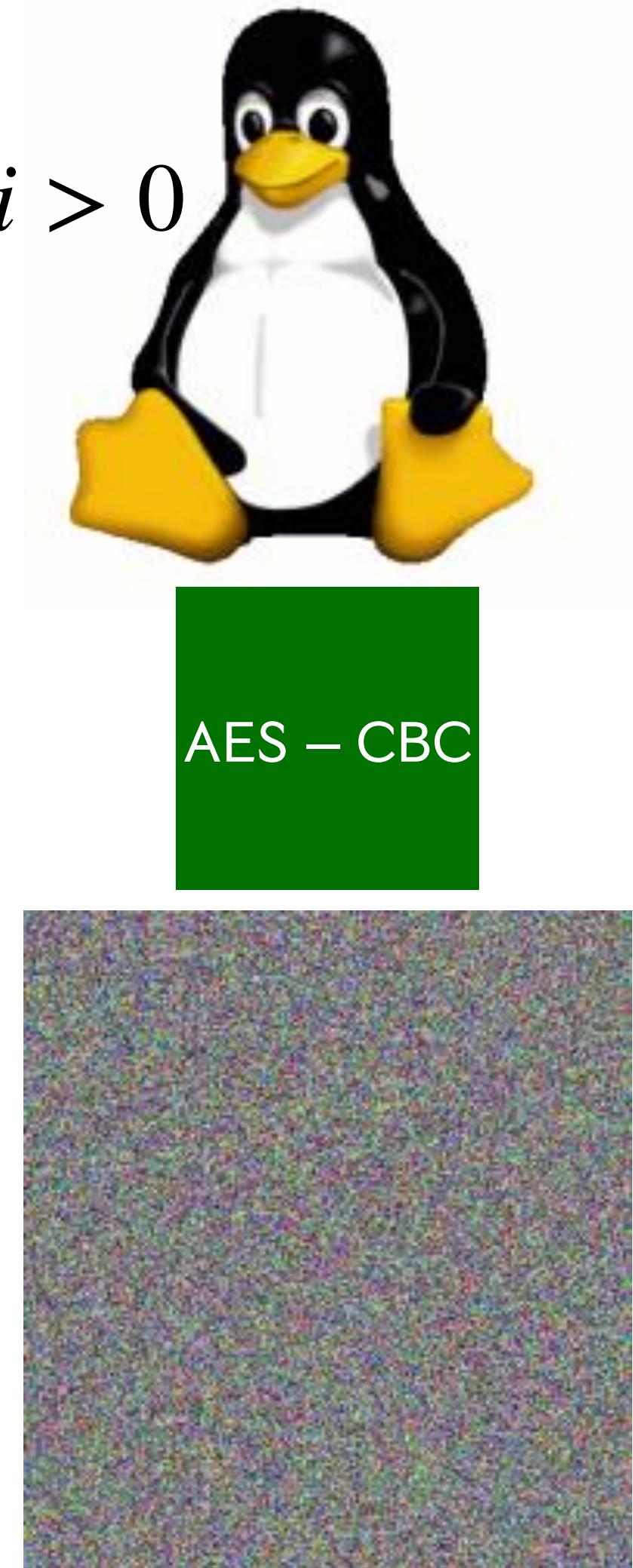
$$m_0 = D(k, c_0) \oplus IV$$

$$m_i = D(k, c_i) \oplus c_{i-1} \text{ for } i > 0$$



IV is the source of randomness

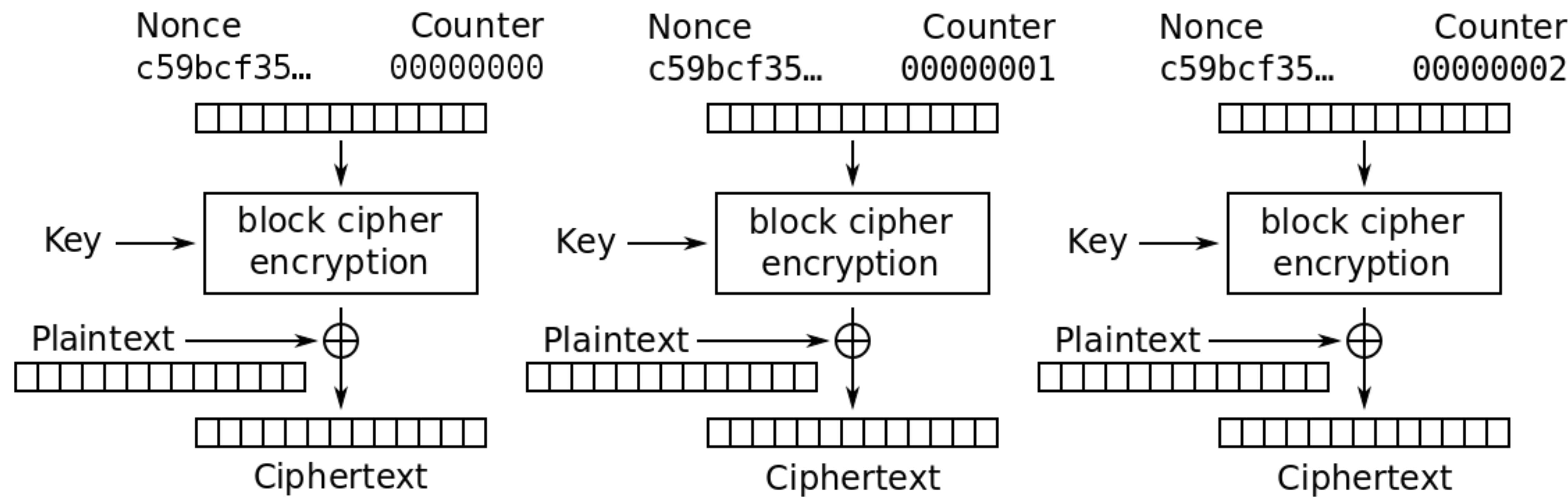
IV needs to travel with the ciphertext to enable the decryption of c_0



Counter Mode (CTR)

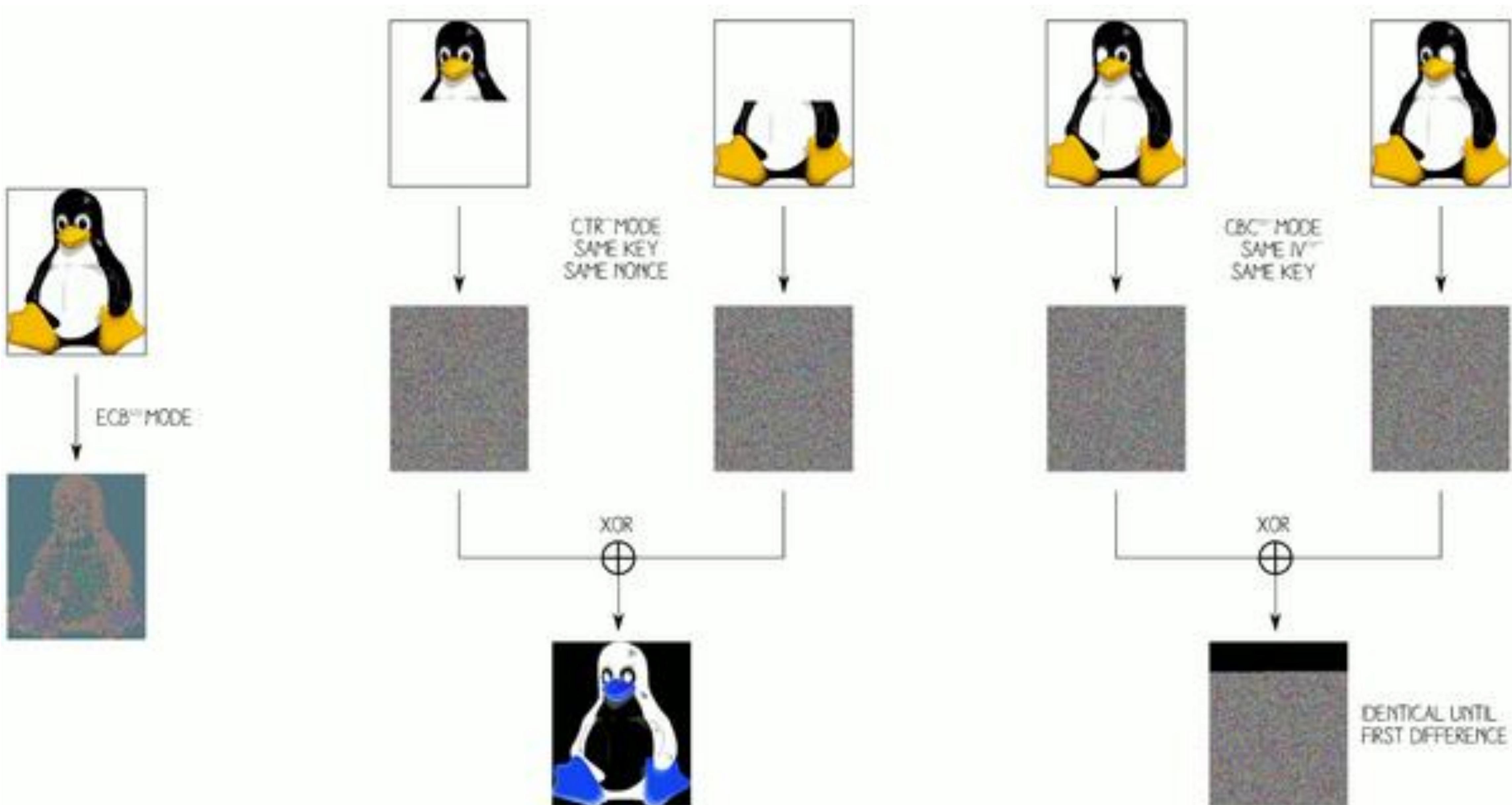
- + Encryption and Decryption are parallelizable (and basically E=D!)
- The nonce value is needed to decrypt every ciphertext block

CTR is OK to use, with care: nonce = number used once...otherwise...

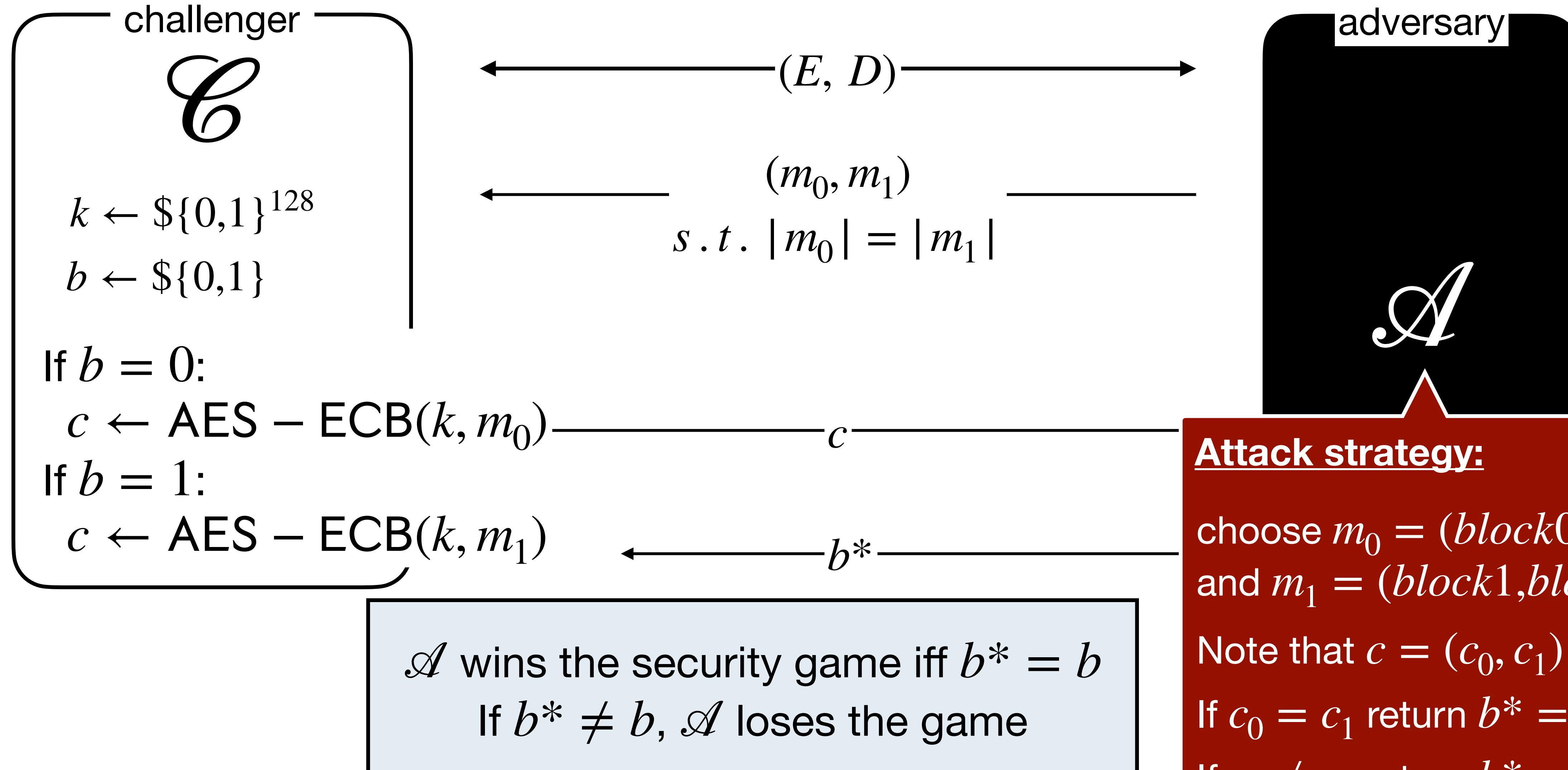


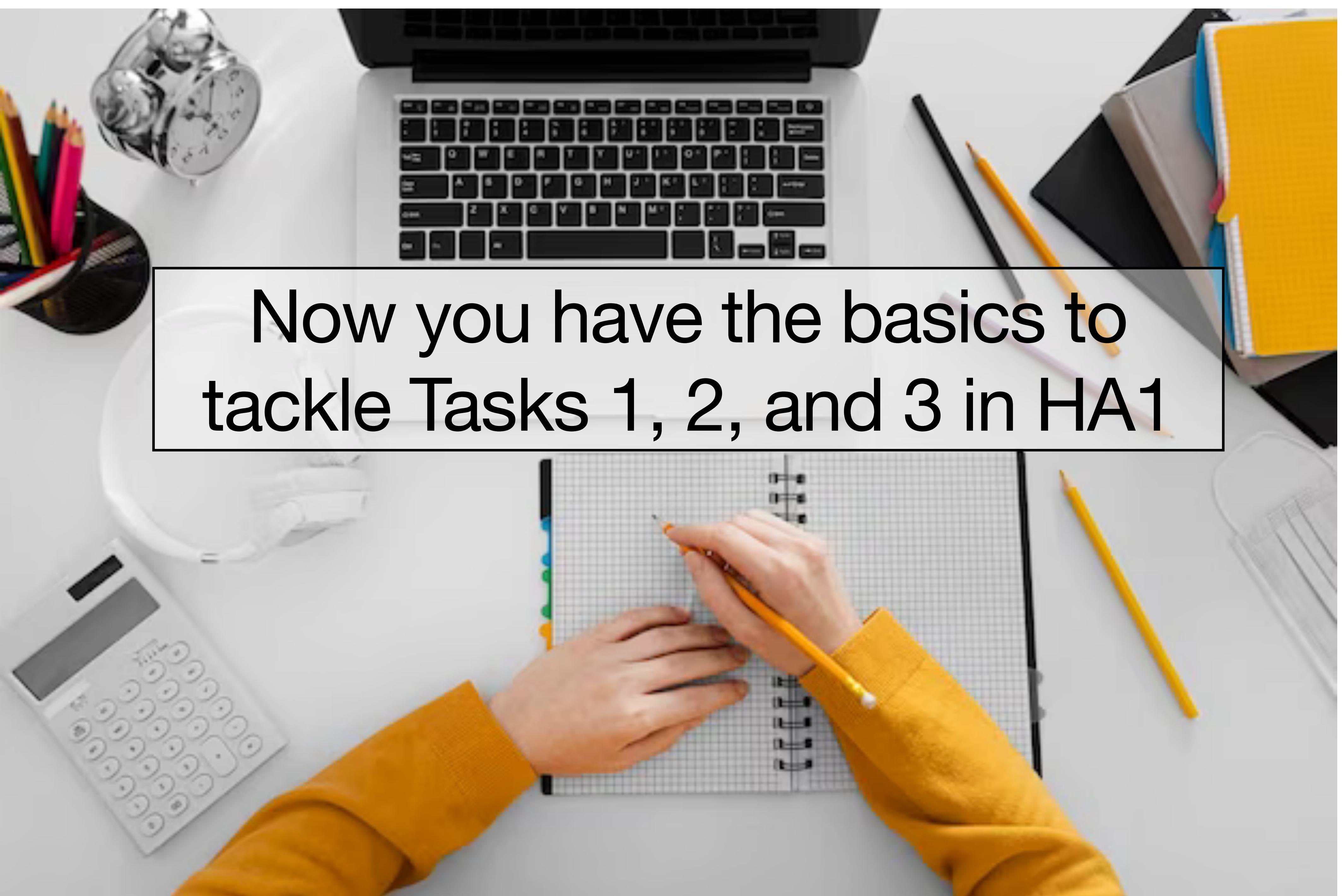
Counter (CTR) mode encryption

Modes of Operation's Failures - Visual Examples



Is AES-ECB Semantically Secure?





Now you have the basics to
tackle Tasks 1, 2, and 3 in HA1

Cryptography (TDA352 / DIT352) - Lecture 3

Blockchain and related technologies

Victor Morel

Chalmers University of Technology

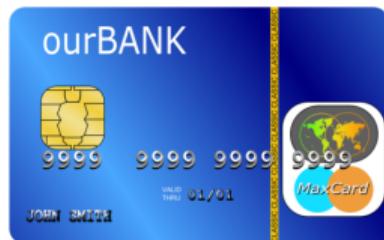
morelv@chalmers.se



November 12, 2024

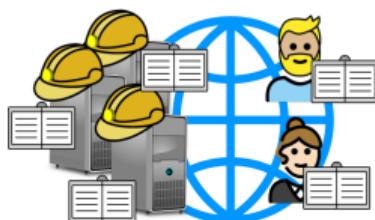
Previously in cryptography ...

Currencies over the millennia



Why this trend? 🤔

Cryptocurrencies and trust



Classic situation

We trust the bank (centralised & fully trusted)

What if we don't trust the bank? 🤔

Replace the bank with a wide collection of anonymous and autonomous parties (distributed system). Why? 🤔

What do these parties need to do?

- › define what a valid transaction is
- › agree on a history of transactions (ledger)

Definition of a blockchain

A blockchain is a ledger 

Three important properties:

- Distributed ✓
- Immutable
- With a consensus mechanism

The original blockchain: Bitcoin



Transactions

Transactions are composed of

- ID
- signature for ownership
- inputs
- outputs

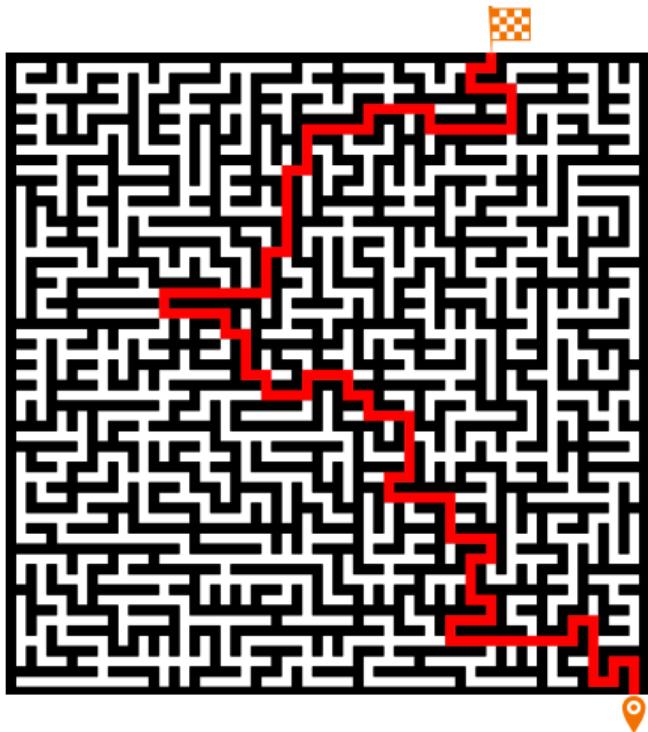
All transactions are public, only need to verify the signature.

Incentives

Transaction fee for incentive to mine new blocks.

Block reward to mine new blocks by solving a cryptographic puzzle.

Cryptographic puzzles



A cryptographic puzzle is like a labyrinth

- › Well-defined problem
- › Hard to find a solution
- › But easy to verify it

Used in:

- › Transferring coins (combined with digital signatures, see Module 2)
- › Coin mining (minting)

Which puzzles are used in Bitcoin?

Cryptographic hash puzzles - intuition



It's like a smoothie

Given a smoothie, it's hard to find the exact ingredients and proportions. But if you have the ingredients and proportions, making the smoothie amounts to pressing the "on" button!

In other words

- Given $y \in \{0, 1\}^d$
- Find an input value $x \in \{0, 1\}^n$
- Such that $H(x) = y$

But also: H is a cryptographic hash functions, i.e., it is designed to be easy to compute, but computationally hard to "invert" (finding a pre-image of a given value y)

Cryptographic hash puzzles - cont'd

The Bitcoin Hash Puzzle (Proof of Work)

- › Let $y = (y_L, y_R)$ and $x = (x_L, \textcolor{red}{x}_R)$
- › Given $y_L = 0^{19}$ $\textcolor{violet}{y}_L = 0^{19}$ and x_L , find $\textcolor{red}{x}_R$ s.t. $\text{sha256}(x_L, \textcolor{red}{x}_R) = (\textcolor{violet}{y}_L y_L, y_R)$
- › Find a **nonce** value that makes a given input hash into an hex string with **19 leading zeros** 19 leading zeros.

How do we adjust the hardness? 🤔

$\text{sha256}(\text{I love Crypto!}) =$

e8f6178df67ea4ec791b9fd72a2d710a3d832c113ee933a0654ae0e423d49ac9

$\text{sha256}(\text{I love Crypto!-}\textcolor{red}{251509386766}) =$

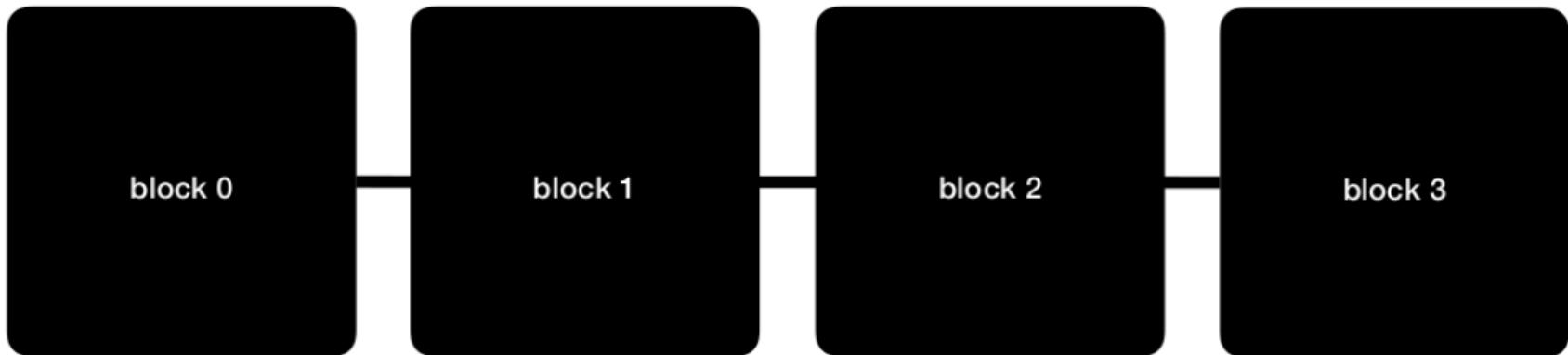
0000092273023b5bc71c29852a01d0121336c16e700535cca2a8c5ef1459becd

This is basically Bonus Assignment 1 😊 (here with 5 leading zeros)

I want to build a snowman I want to build a ~~snowman~~ ledger

The basic idea is to link blocks of data

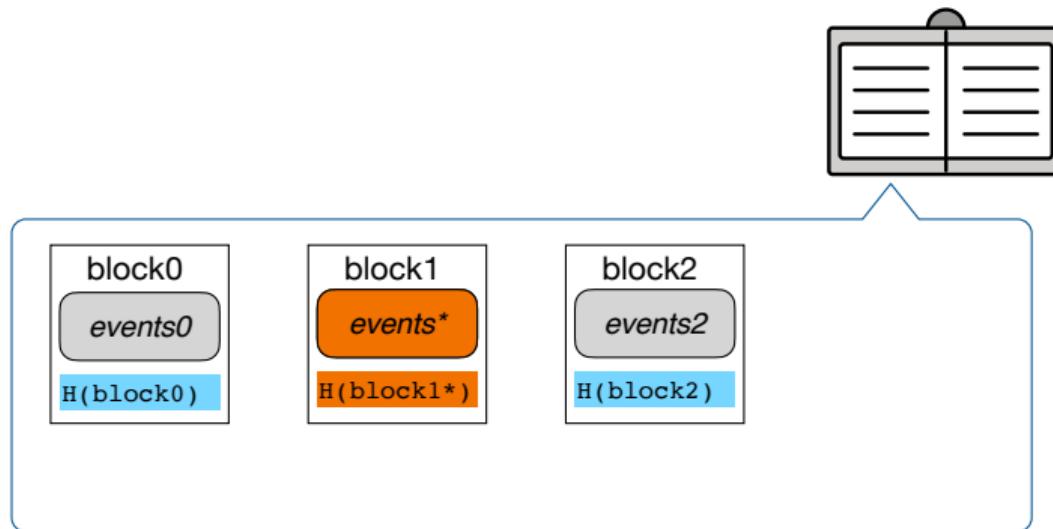
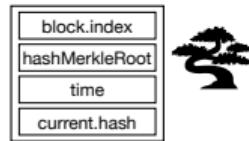
- › Blocks partition time into epochs / periods / time windows
- › Anything that happens in one time period is recorded into a block
- › One block corresponds to several transactions
- › Any change to an 'old' block affects all following blocks



I want to build an immutable ledger I

Attempt 1

Block Structure



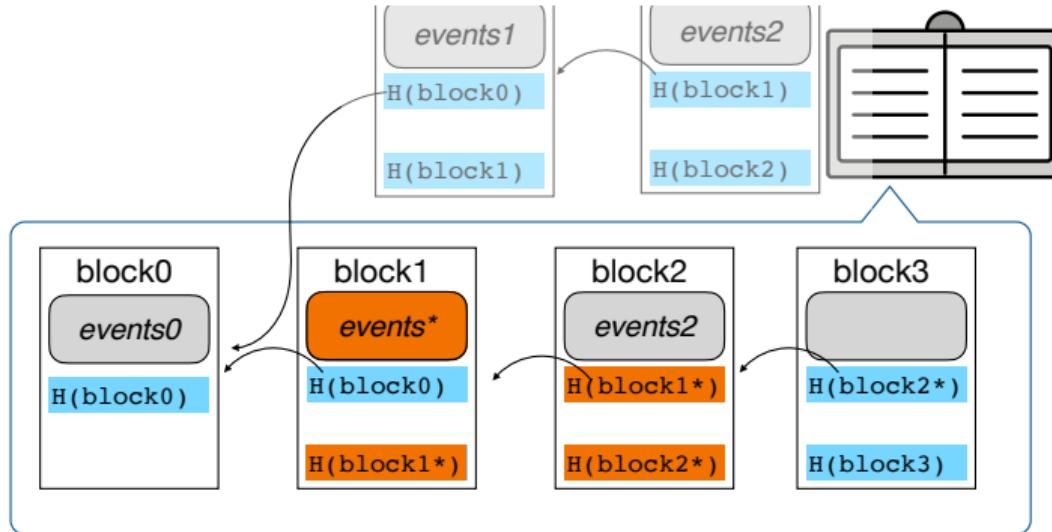
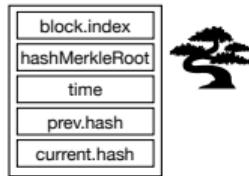
What are the issues? 🤔

- Only blocks, no chain!
- No proof of work (anyone can mine a block)
- No immutable history of transactions

I want to build an immutable ledger II

Attempt 2

Block Structure



Now we got the chain  But is the past really immutable? 

The hash of a tampered block can be computed efficiently!

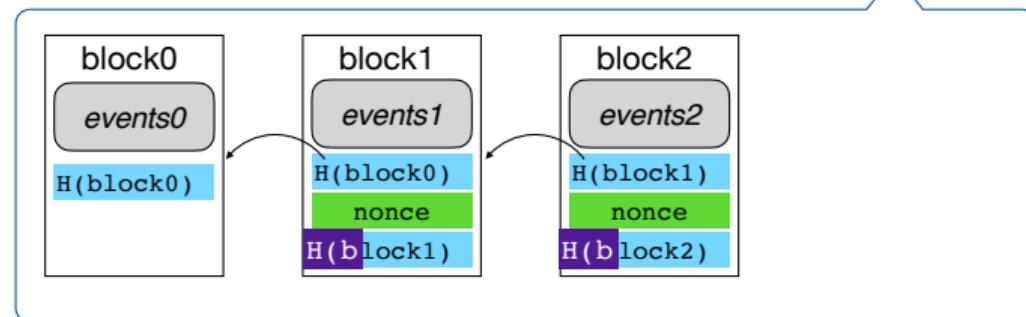
Anyone can (easily) mine a block 

I want to build an immutable ledger III

Final Attempt

Block Structure

block.index
hashMerkleRoot
time
prev.hash
nonce
current.hash



Find a **nonce** value that makes the hash of the current block start with **19 leading zeros**.

Definition of a blockchain II

A blockchain is a ledger 

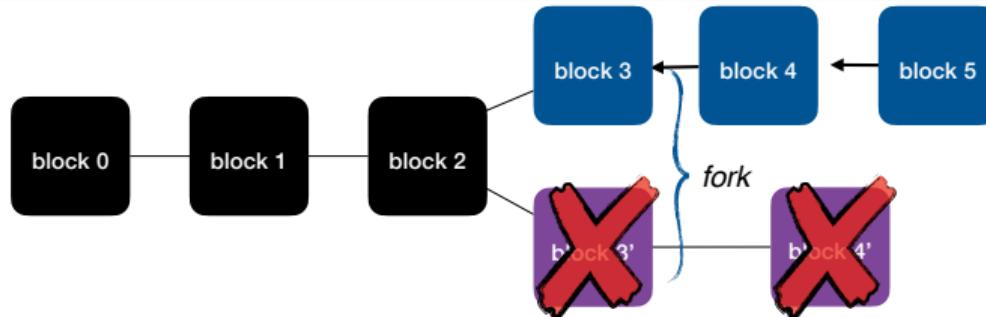
Three important properties:

- Distributed ✓ (set of autonomous parties, no SPOF)
- Immutable ✓ (cannot rewrite history)
- With a consensus mechanism

Ledger rules

The rules

- › The past is immutable
- › Everyone agrees on the history (consensus)
- › Always build on the longest branch (longest chain rule)
- › Each block hash needs to start with a given prefix (lower the chance that blocks appear at the same time, uses the proof of work)



If multiple branches live, then you can spend your money twice! (double-spending)

Consensus in Bitcoin



miners vote for the correct chain by mining along it
(longest chain \Rightarrow highest amount of votes = consensus)

Our properties

- › Distributed ✓
- › Immutable ✓
- › With a consensus mechanism ✓

Sybil attack

- › The attacker creates multiple identities to gain influence
- › How does Bitcoin prevent sybil attacks? 🤔
- › Thanks to the proof of work
- › Creating multiple entities does not create additional computational power!

It is feasible to steal the consensus in small blockchains

Bitcoin Gold hit by 51% attacks, \$72K in cryptocurrency double-spent

Smaller Proof-of-Work networks are prone to these incidents

January 27, 2020 - 2:37 pm



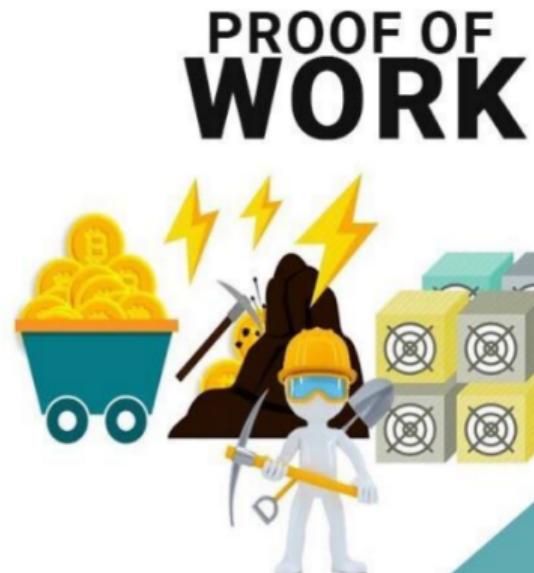
Of the growing ecological impact of Bitcoin

Bitcoin is terrible for the environment – can it ever go green?

Cryptocurrency mining uses huge amounts of energy, but activists are urging for a change in its code to reduce its environmental impact



Another consensus mechanism



VS



PROOF OF
STAKE

Proof of stake I

In a nutshell

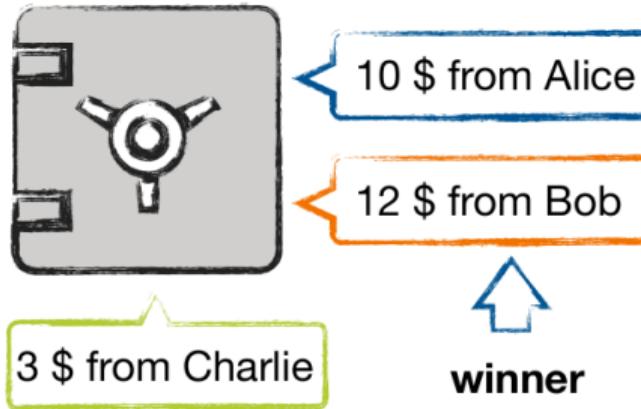
- › Change the rule for selecting the lottery winner
- › Replace “work” with “stake” (computation power with digital cash)
- › The probability of being selected is directly proportional to one’s stake in the “bid”



Proof of stake II



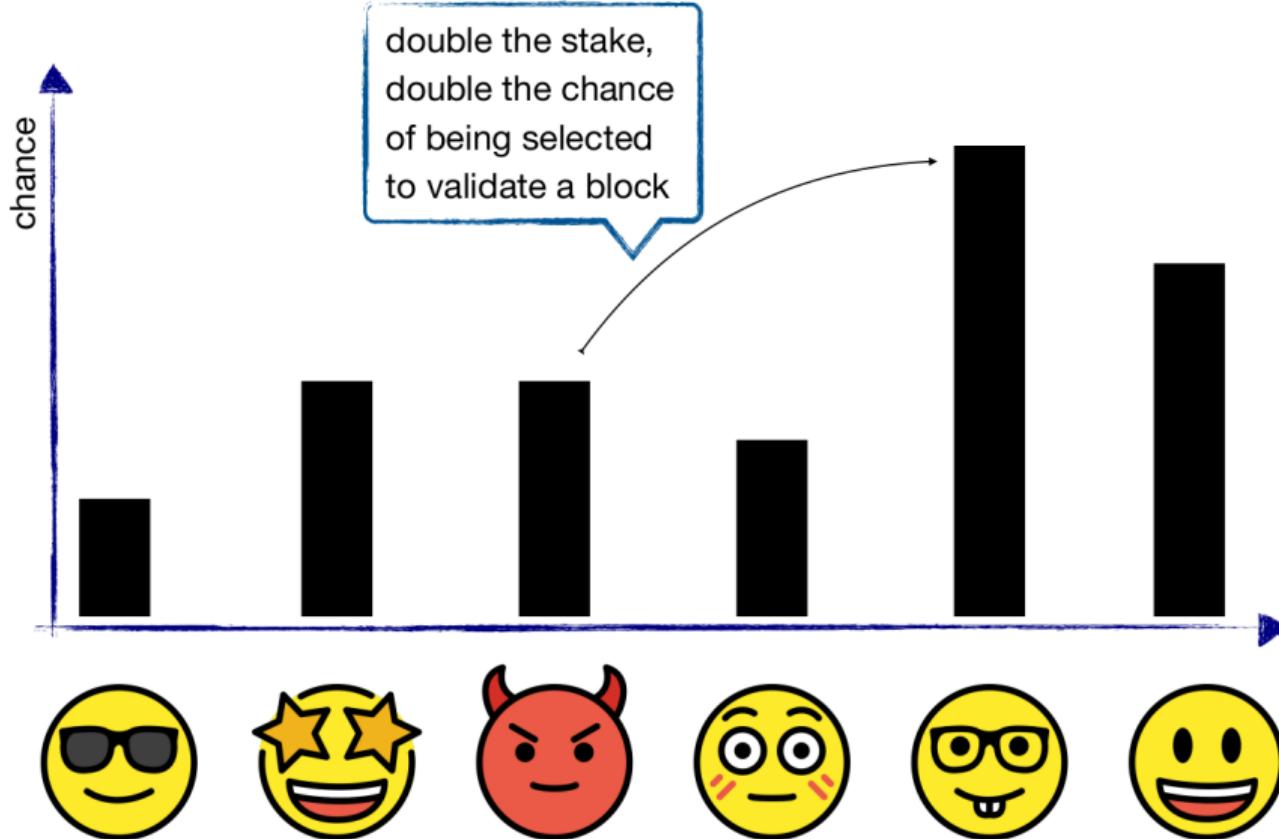
*temporary lock stakes to determine
who mints the next block*



Incentives

- › The winner gets to check if all transactions in the block are valid, earns the transactions fees
- › Validators lose part of their stake if they approve fraudulent transactions

Proof of stake III



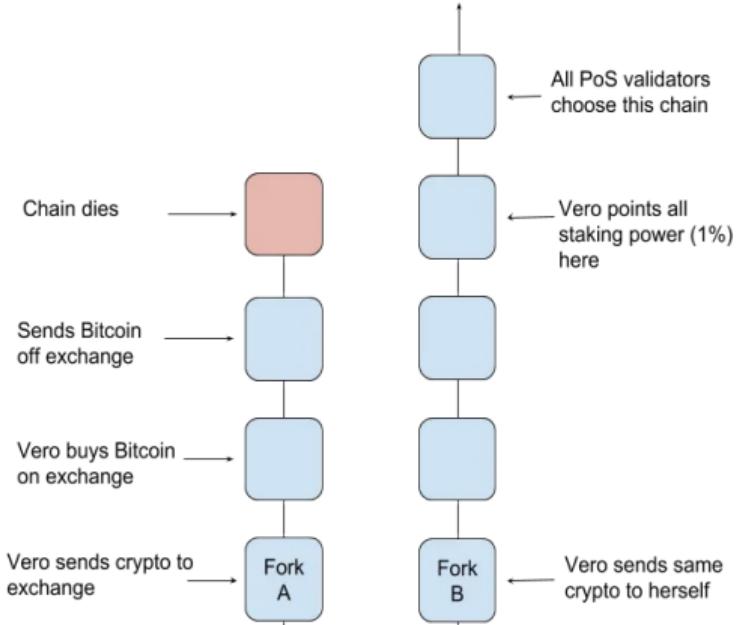
Downsides of PoS

Nothing at stake theory

Considering that:

- › Validators always prioritize profit
- › **No one** is altruistic (will choose only one chain)
- › A fork has occurred and validators are actively building both forks
- › Software is modified to do mine on all forks

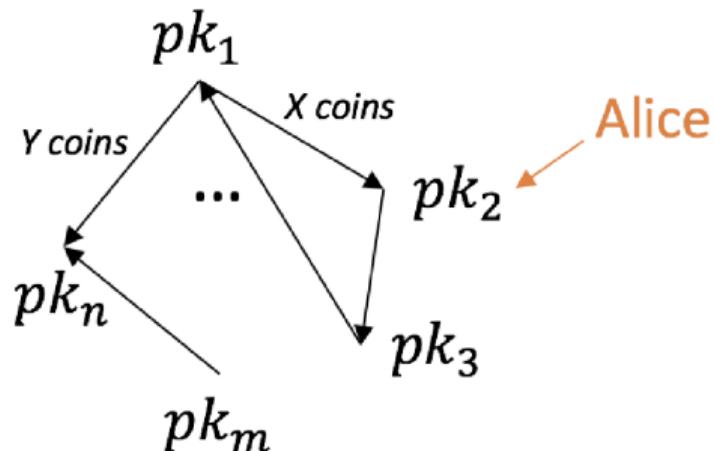
Hard to pull-off in practice 😱



How anonymous is Bitcoin?

Bitcoin is weakly anonymous

- Parties are anonymous in the sense that only public keys are revealed
- One's real identity is therefore hidden in theory
- But the transaction graph is public, it can be used to deanonymize



Zcash provides the possibility to perform "anonymous" (shielded) transactions using Zero-Knowledge Proofs (notion further studied in the course).

Note that many exchanges require a proof of identity to open a wallet, in order to fight fraud.

Are cryptocurrencies entirely solving trust?

That's not that easy ...

- › Blockchains work because we trust the math/crypto behind
- › They operate a shift in the trust from institution to technology
- › But code can have bugs, and wallets can get stolen ... without recourse
- › And one still needs to trust people (e.g. to change the bitcoin block size)

And while the protocol is distributed ...

The technology stack is not:

- › Few exchange places
- › One main company for the mining hardware

Are cryptocurrencies even currencies in the first place?

Not really, according to economists

In economy, a currency is:

- A unit of account, but Bitcoin fluctuates too much ✗
- A store of value, but Bitcoin doesn't have intrinsic value ✗
- A medium of exchange, Bitcoin can do that ✓
- But 1) transaction fees are pricey for small operations and 2) no refund possible

Only a legal tender (official currency) in El Salvador

Better call them **crypto-assets**

Hash functions



Back to our smoothy

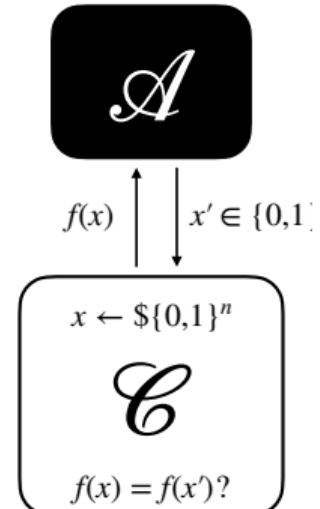
- “Easy” to compute
- “Hard” to invert
- Easy to blend, hard to figure out the ingredients and proportions
- Here, easy and hard are to be understood according to computational complexity (polynomial time)

One-way functions (OWF)

Definition

A function $f : \{0, 1\}^n \rightarrow \{0, 1\}^d$ is one-way if:

- › There exists an algorithm that computes $f(x)$ in polynomial time for all inputs $x \in \{0, 1\}^n$ (f is efficiently computable)
- › For every PPT algorithm \mathcal{A} there is a negligible function $negl_{\mathcal{A}}(\cdot)$ such that for sufficiently large values of $n \in \mathbb{N}$ it holds that
$$Pr[f(x) = f(x') | x \leftarrow \$\{0, 1\}^n, x' \leftarrow \mathcal{A}(f(x))] \leq negl_{\mathcal{A}}(n)$$
With | conditional probability



Constructing one-way functions

Example: OWF from integer factorisation

Consider $f : \{k\text{-bit primes}\} \times \{k\text{-bit primes}\} \rightarrow \mathbb{N}$, for $k = 256$, defined as: $f(p, q) = p \cdot q$.
 $f(\cdot)$ is a one-way function if integer factorisation is (computationally) hard.

What happens if we consider $f : \text{primes} \times \text{primes} \rightarrow \mathbb{N}$, $f(p, q) = p \cdot q$? 🤔

OWF - security note

OWF only guarantee that the input x is not leaked *entirely*.

This means that it is still possible that $f(x)$ leaks a substantial amount of information about x .

Example

Let $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a OWF.

Consider the function $g : \{0, 1\}^{2n} \rightarrow \{0, 1\}^{2n}$ defined as $g(x_0||x_1) := f(x_0)||x_1$.

Even if $g()$ reveals half of its input, it is still a OWF!

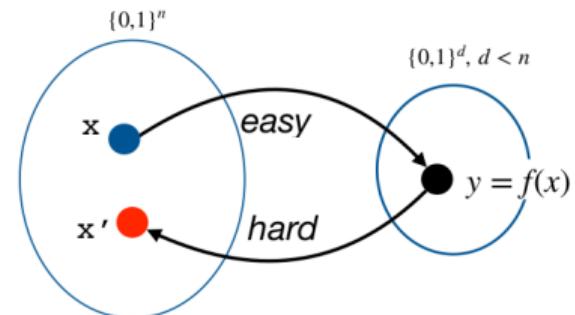
Why? 🤔 Because f is a OWF, see proof on board.

A special kind of OWF - Cryptographic Hash Functions

Definition

A cryptographic hash function $H : \{0, 1\}^n \rightarrow \{0, 1\}^d$ has the following properties:

- › one-way (i.e. pre-image resistance) $\Pr[H(x) = H(x') | x \leftarrow \$\{0, 1\}^n, x' \leftarrow \mathcal{A}(f(x))] \leq \text{negl}_{\mathcal{A}}(n)$
- › compressing (i.e., $n > d$) and of fixed output length d
- › collision resistant (i.e. second pre-image resistance)
 $\Pr[H(x) = H(x') | x, x' \leftarrow \mathcal{A}(H), x \neq x'] \leq \text{negl}(n)$



Does this mean that there exist **no** collision for hash functions? 🤔

Note that cryptographic hash functions are widely used to store an obfuscated version of a piece of information in a deterministic way (e.g. passwords).

What are the odds of two people having the same birthday?



Can we estimate the probability of finding a collision?

The intuition

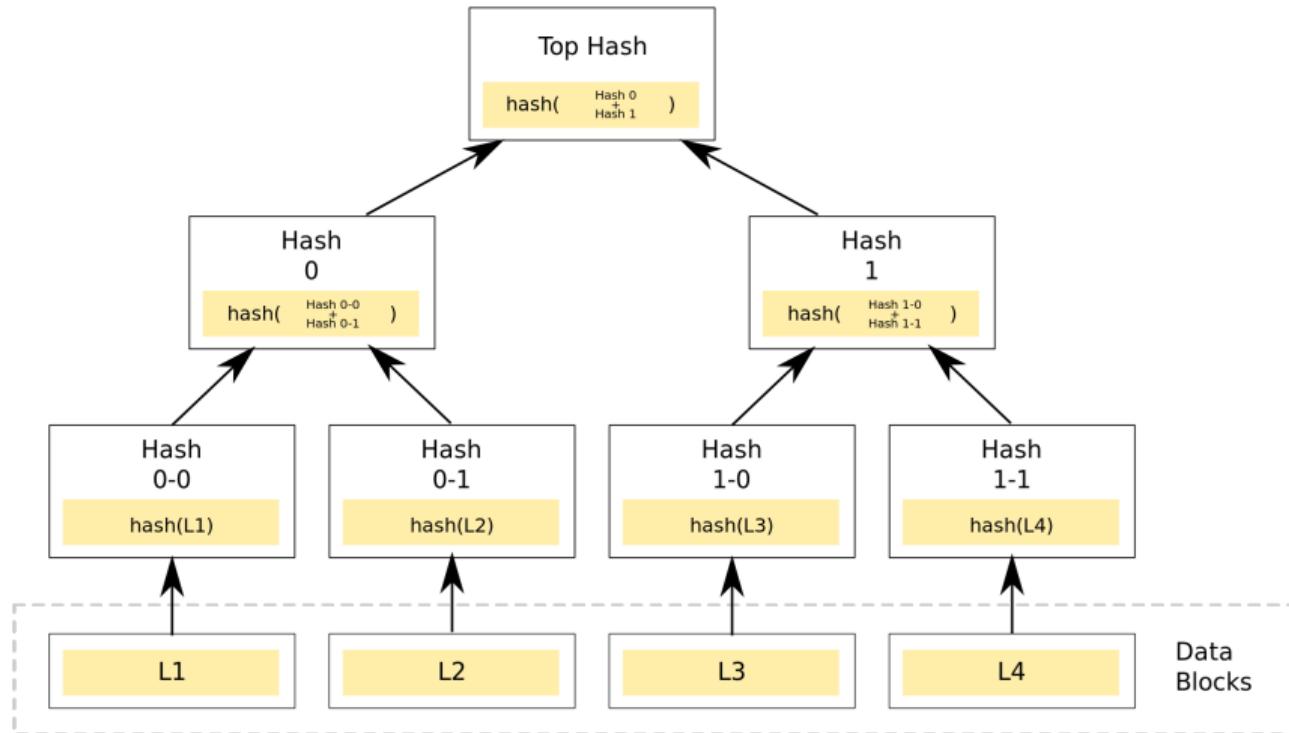
- › 365 days a year, $1 - \frac{1}{365} = \frac{364}{365}$ probability that a given pair of birthdays are different
- › For n people, there is $\binom{n}{2} = \frac{n(n-1)}{2}$ possible pairs
- › The probability of having n different birthdays is like “throwing the birthday dice” as many times as you have pairs, or $(\frac{364}{365})^{\frac{n(n-1)}{2}}$ **not** to have a collision (independent events)
- › Now we calculate the complement to have the probability of a collision $1 - (\frac{364}{365})^{\frac{n(n-1)}{2}}$

For hash functions

- › We consider a function $H(\cdot)$ acting as purely random function (Random Oracle Model)
- › Days a year \rightarrow size of the hash digest N ; people in a room \rightarrow attempts to find a collision
- › With a large enough output domain, it is unfeasible in practice (see sha256)

See **prob_bParadox.pdf** on Canvas for a proof of the bounds on the probability to have collisions.

Merkle trees - binary hash trees

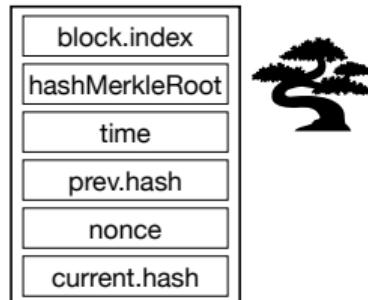


Usages of merkle trees *in the wild*

In blockchains

- › Merkle trees are used to store data about transactions
- › Instead of storing all transaction data, only the Merkle root is stored in a block

Block Structure



Remember?

On the web

- › In Certificate Transparency
- › Certificate Transparency is a security standard
- › Used to monitor and audit digital certificates of websites
- › Certificates are stored in a public and distributed ledger, the structure of which is an append-only Merkle tree

Also used in this tiny software called **git** . . .

Do I need a blockchain or a merkle tree?

Blockchain can be heavy

- They are needed in specific situations
- Their implementation cost can otherwise be prohibitive
- And the prevalence of the Proof of Work has significant ecological impact 

Sometimes, all you need is a Merkle Tree

To certify some documents such as diplomas, a blockchain is not required:

- Diplomas  are official documents certified by trusted third parties (e.g. Chalmers)
- There can be an interest in maintaining a distributed ledger of diplomas
- In this context, not everyone should be able to write in the ledger
- Also, a consensus mechanism is not required
- Therefore, a Merkle  is sufficient

Connections with assignments

HA1

What you learned about the Birthday Paradox will help you solve Tasks 5 and 6 of HA1.

Bonus 1

What you learned about mining Blockchains will help you solve Bonus 1. (A little bird told me someone already has a 

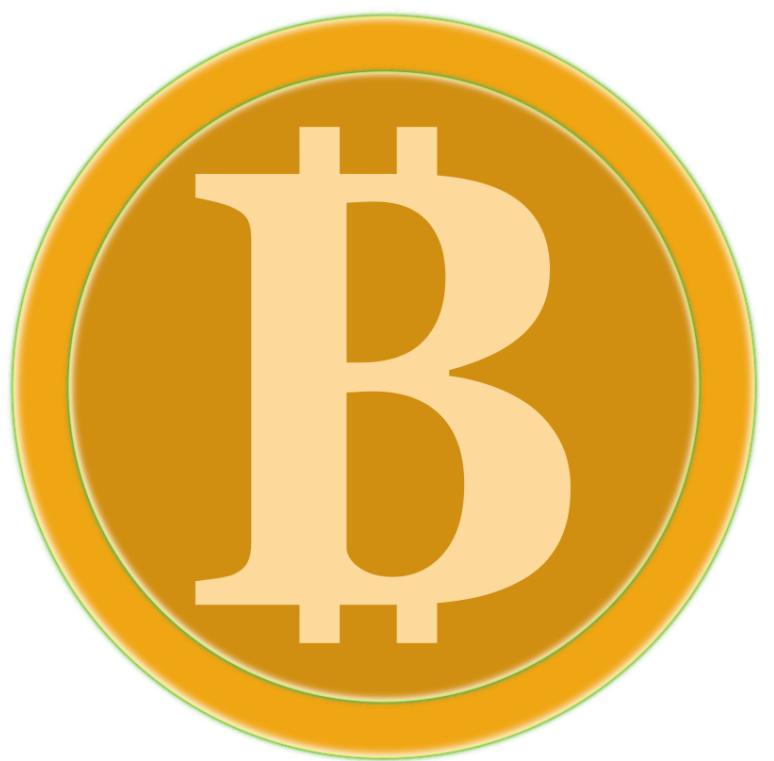
Feedback form for the lecture



HA1 First Hand in Deadline Is Next Tuesday

If you have doubts, use the **Office Hours 11:30-12:30 on Monday(s) in EDIT 6128**.
The PhD TA responsible for grading HAs will be there happy to answer your questions!
Do not ask me/the lecturer about tasks in HAs.

Recap From the Last Lecture



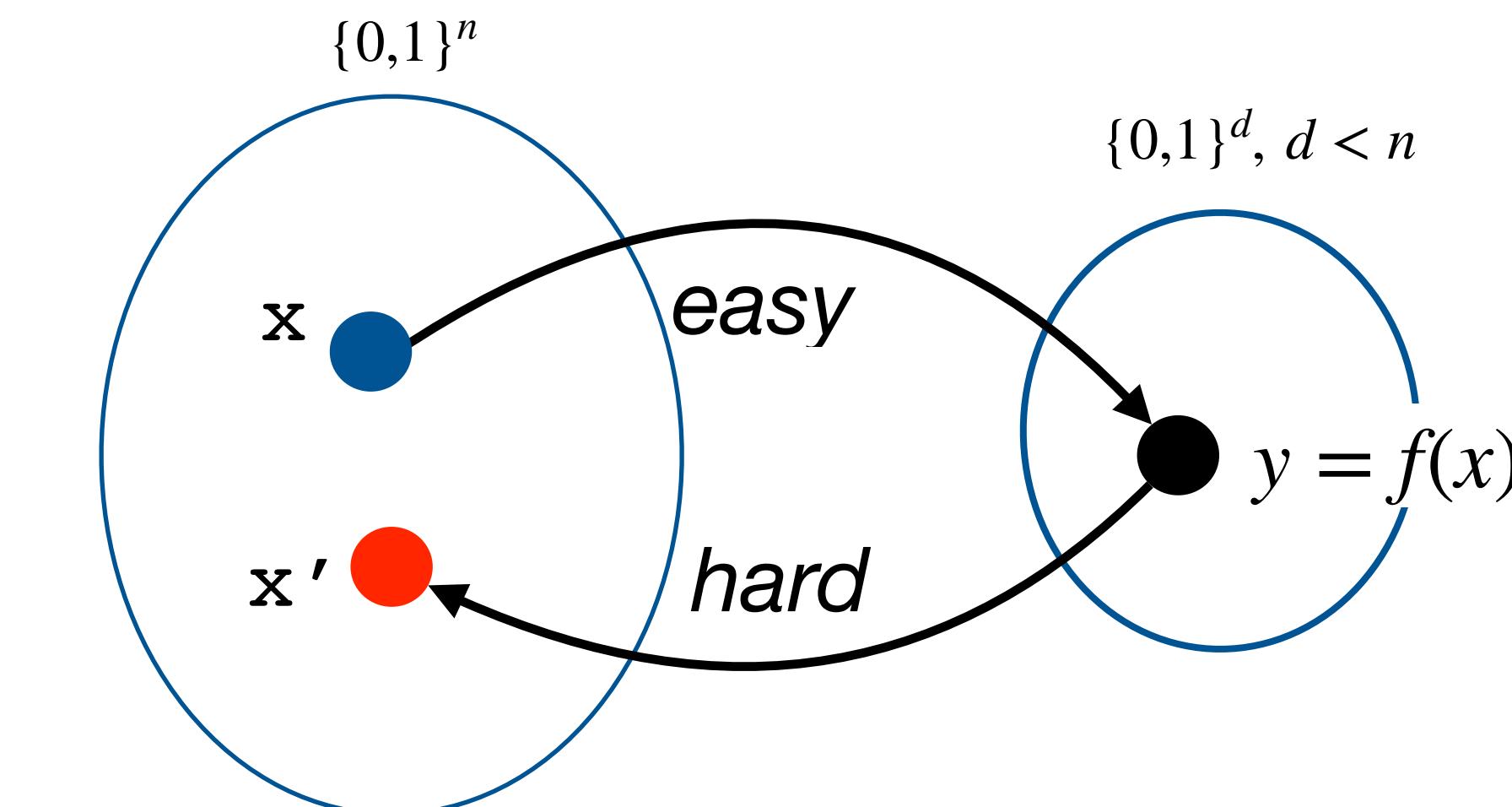
OWF



Cryptographic
HASH
Functions

pre-image resistance
&
second pre-image res.

Merkle Trees



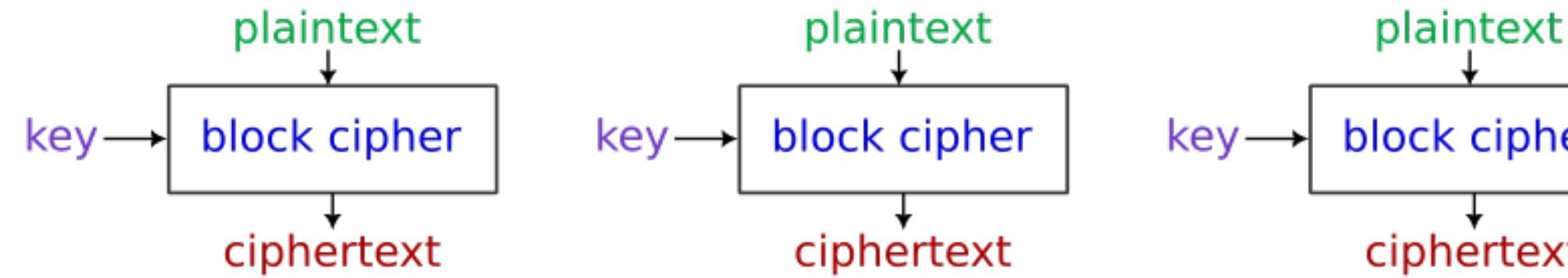
Birthday Paradox

PoW vs PoS

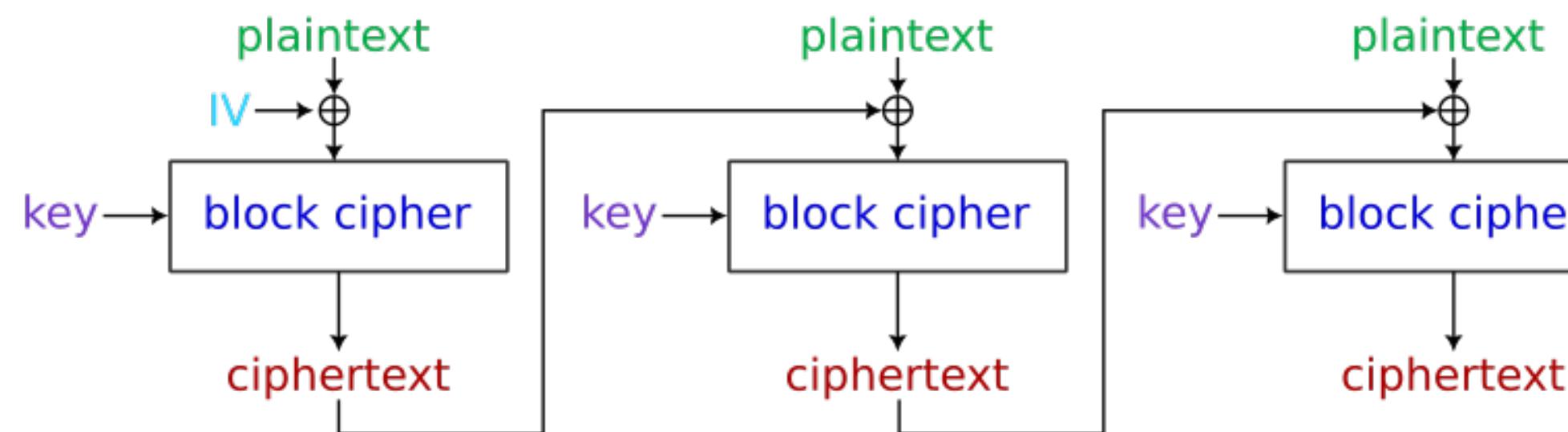
Sybil Attack

Recap From Lecture 2

Electronic codebook (ECB)



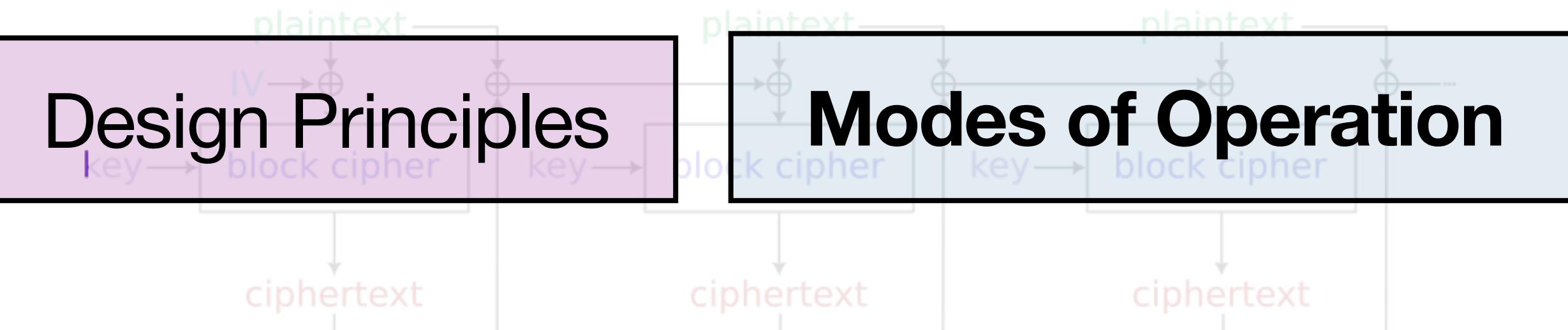
Cipher block chaining (CBC)



PRP / Block Ciphers

AES

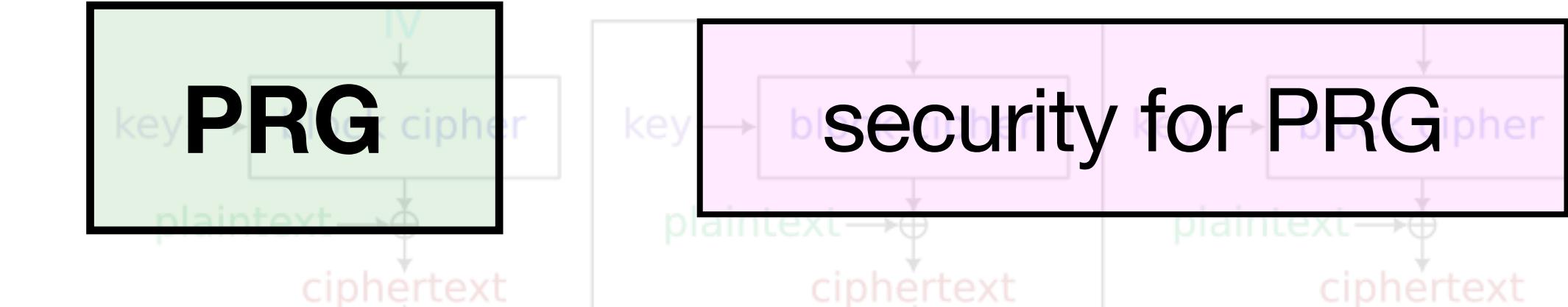
Propagating cipher block chaining (PCBC)



Design Principles

Modes of Operation

Cipher feedback (CFB)



semantic security

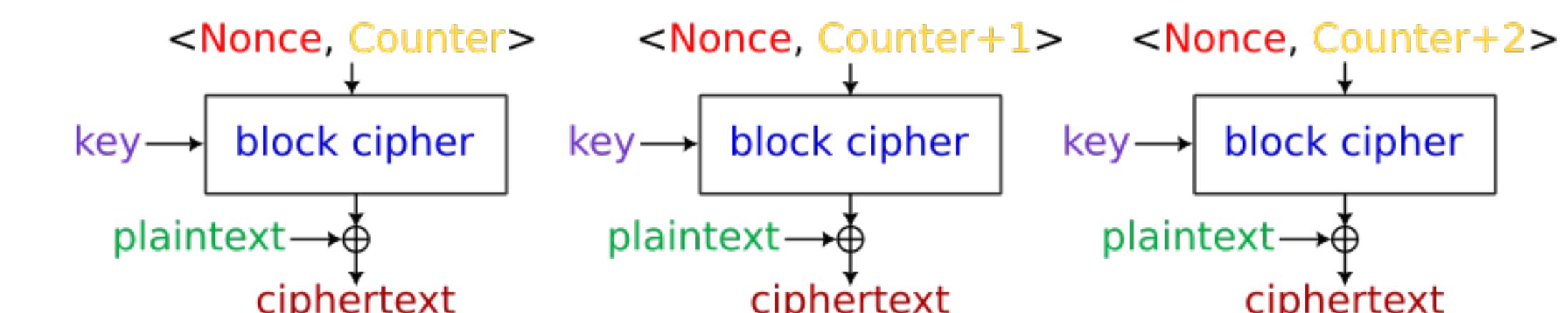
negligible

PRG-OTP is semantically secure

Multi-message security?



Counter (CTR)



Lecture 4 - Agenda

Secure Communication

- Towards IND-CPA Security
- Why Does Integrity Matter?

Message Authentication Codes

- Definition + Unforgeability
- CBC-MAC attack
- MAC vs Hash Functions
- HMAC

Authenticated Encryption

- Encrypt-And-MAC
- Encrypt-Then-MAC
- Galois Counter Mode (GCM)

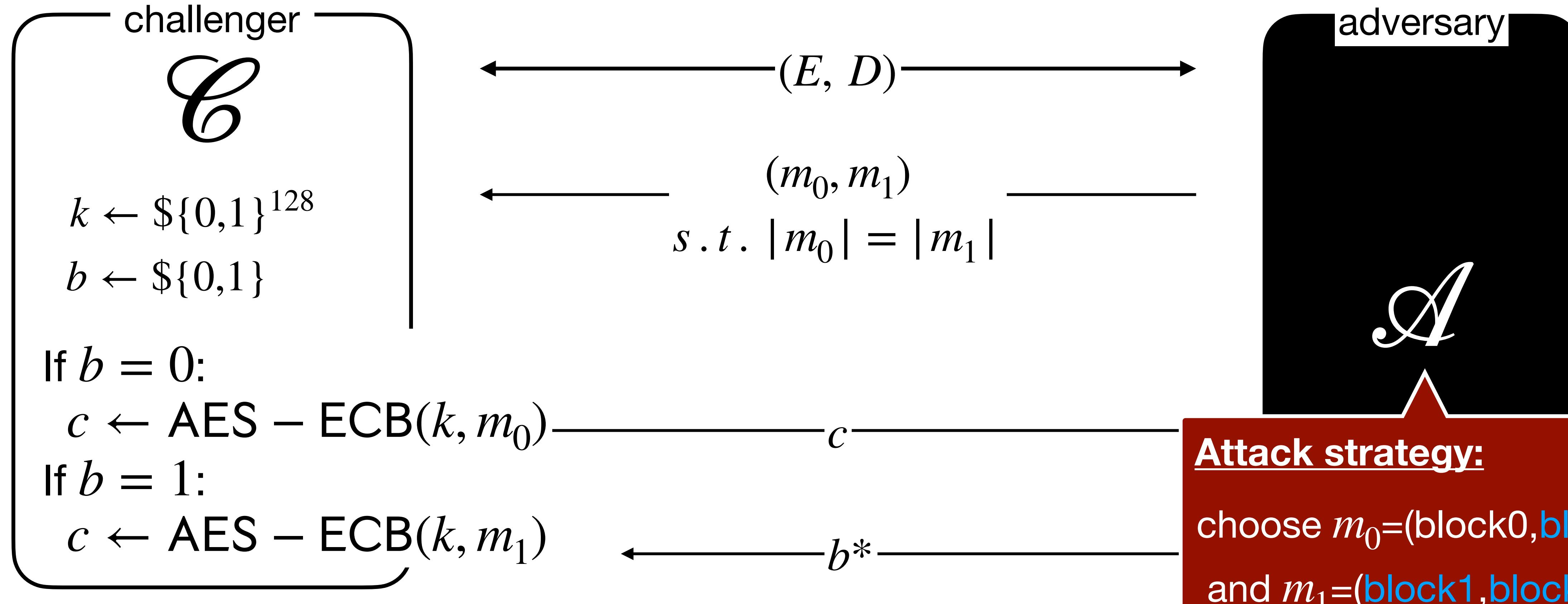
Overview of Module 1

Exercises From Exam

Towards IND-CPA Security

**there are 10 kinds of
security games in the world of encryption**
the RoR game and the LoR game
those who understand binary
and those who dont.

Is AES-ECB Semantically Secure?



\mathcal{A} wins the security game iff $b^* = b$
If $b^* \neq b$, \mathcal{A} loses the game

Attack strategy:
choose $m_0=(\text{block0}, \text{block1})$
and $m_1=(\text{block1}, \text{block1})$
Note that $c = (c_0, c_1)$
If $c_0 = c_1$ return $b^* = 1$
If $c_0 \neq c_1$ return $b^* = 0$

A photograph of two orange cats in a lush green garden. One cat is in the foreground, looking towards the right, while the other is partially visible behind it. A white text box is overlaid on the image, containing the question.

Are other modes of operation “secure”?

Yes! and they also satisfy a new, stronger security notion

Lessons Learned So Far



From ECB: **deterministic encryption is not semantically secure.**
(if the same key is used to encrypt the same block, the resulting ciphertext will always be the same)

CBC and CTR use **randomness** (IV and nonce respectively), so two encryptions of the same plaintext under the same key will (generally) produce two different ciphertexts

We Want *Randomized (Aka Probabilistic)* Encryption!

Bad news: probabilistic encryption generates ciphertexts **larger** than the plaintext
(this is an inevitable price to pay for *better* security)

Good news: in certain settings we can get good security **without ciphertext expansion**

Towards a New Security Notion

A

Adversary's Goal

~~To decrypt a ciphertext~~

Security can be damaged with much less

~~To gain some information about the plaintext concealed in the ciphertext~~

Vague, we would need to quantify this leakage (possible but..)

To *distinguish* between the encryption of two *known* plaintext messages

In crypto jargon: **indistinguishability under chosen plaintext attack (IND-CPA)**

IND-CPA has many equivalent notions, read [here](#) if you're interested

Historical example: British military would place mines in particular locations hoping Germans would send encrypted messages about that location.

Modern example: Attacker-controlled Javascript on a web page causes victim web client to make a HTTPS connection.

Security Notions for Block Ciphers



Adversary's Goal

To *distinguish* between the encryption of two *known* plaintext messages

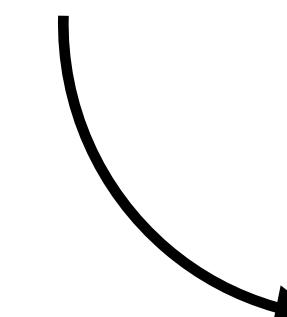
In crypto jargon: **indistinguishability under chosen plaintext attack (IND-CPA)**

Adversary's Power

Efficient algorithm (probabilistic, and runs in polynomial time $< 2^{60}$)

\mathcal{A} can see everything transmitted over the communication channel

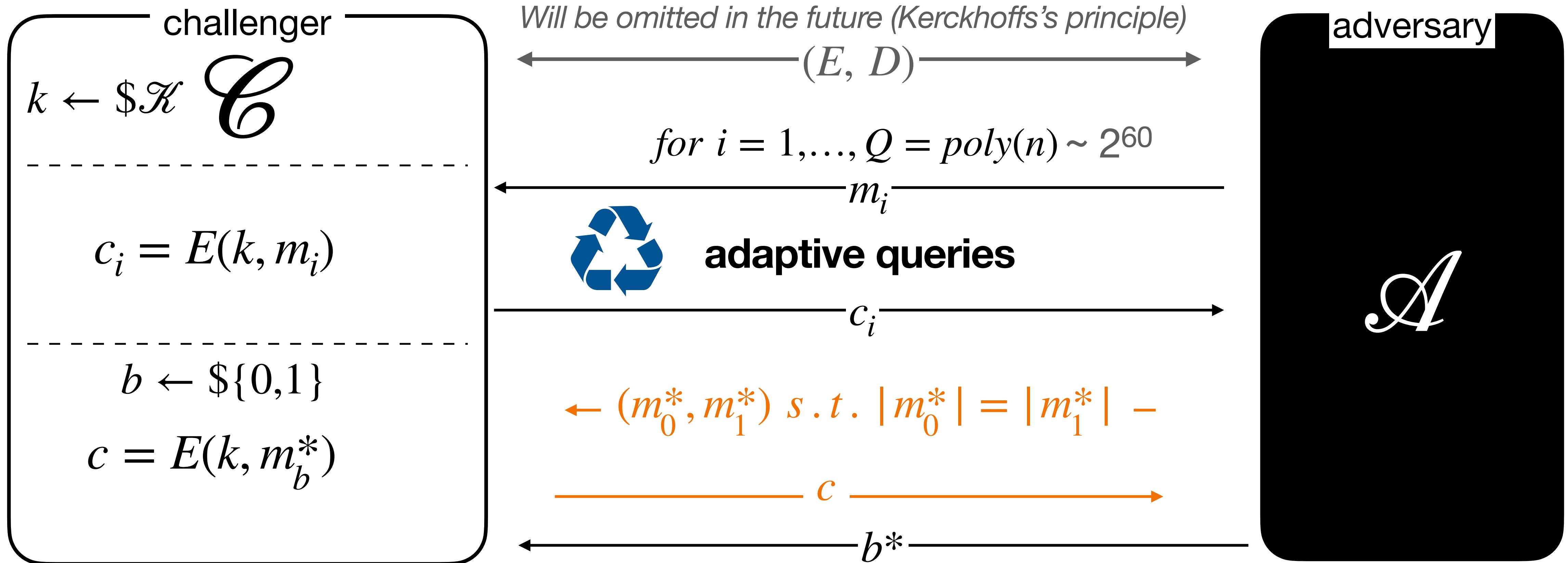
\mathcal{A} knows all details of the encryption scheme except for the secret key
(Kerckhoffs's principle)



“Security through *obscurity*” is an *obsolete mantra*
[computers are good for reverse-engineering, hackers are clever]

IND-CPA (Indistinguishability Under Chosen Plaintext Attack)

Aim: quantify the \mathcal{A} 's likelihood in distinguishing between encryptions of two messages, given that \mathcal{A} can see encryptions of any messages of its choice.



\mathcal{A} wins the security game iff $b^* = b$. If $b^* \neq b$, \mathcal{A} loses the game

Indistinguishability Under Chosen Plaintext Attack (IND-CPA)

Definition: IND-CPA Advantage

An encryption scheme is said to be indistinguishable under chosen plaintext attack (IND-CPA secure) if for any PPT adversary \mathcal{A} that engages in the IND-CPA game only has negligible advantage in winning:

$$Adv(\mathcal{A}) = \left| Pr[\mathcal{A} \text{ wins}] - \frac{1}{2} \right| < negl(n)$$

Is AES-CBC or AES-CTR Mode IND-CPA Secure?

Yes, both are! But we'll skip the formal proofs (too technical for this course)

I didn't formally define what a secure block cipher is... for the purpose of this course this corresponds to (E,D) being indistinguishable from a random permutation over the block space $\{0,1\}^n$

If (E,D) is a **secure block cipher**, then:

- using (E,D) in **CBC** mode yields an **IND-CPA** secure cipher
- using (E,D) in **CTR** mode yields an **IND-CPA** secure cipher

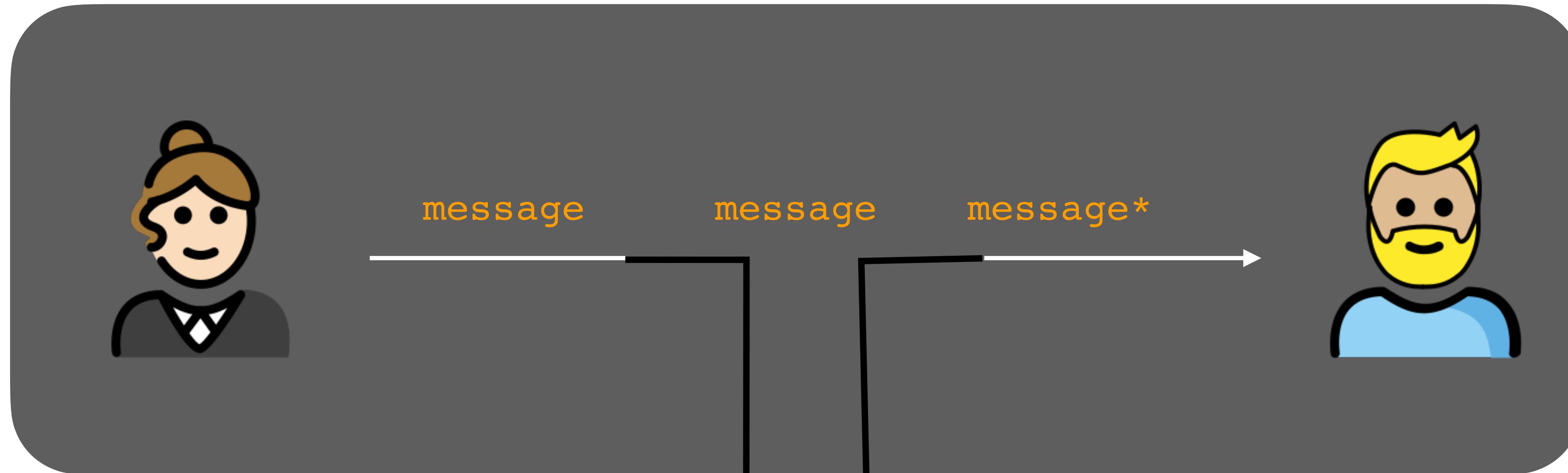
It is not possible to mathematically prove a block cipher to be secure, instead, confidence on its security builds up over years of scrutiny

A Word on Padding for Block Ciphers

Padding of messages is often needed before encryption with a block cipher.
(A message must be as long as a integer number of full blocks)

- Essential property: padding must be **reversible**, i.e., the receiver must be able to remove padding in a unique way.
- Example of **insecure** padding: Add necessary number of **zero bytes** to fill last block.
- The receiver should always check that the padding is correctly applied when removing it. In case of mismatch, the protocol should be immediately aborted.

Secure Communication Over an Insecure Channel



Today: \mathcal{A} should **not** be able to **modify** messages in an undetectable way, or to **impersonate** a sender

Integrity / Authenticity

Until now: \mathcal{A} should **not** be able to **distinguish** between the encryption of two **known** messages (**IND-CPA**)

Confidentiality / Privacy

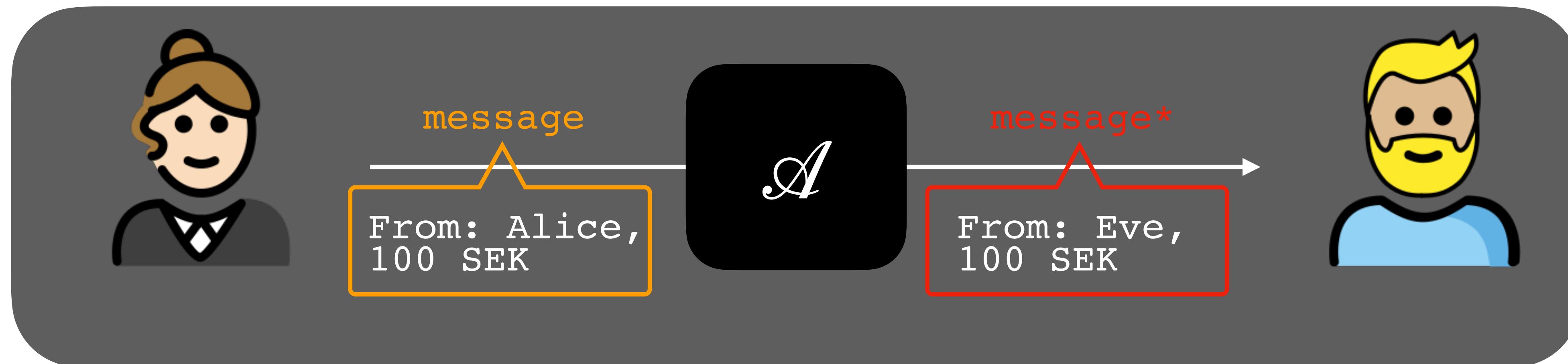
Why Does Integrity Matter?

Topic useful for Task 4 in HA1

A motivating example

Fact1: files sent over a network have well-known, **predictable headers**. A typical example is emails, which have sender (`From:`) and receiver (`To :`) info, as well as date, subject and others.

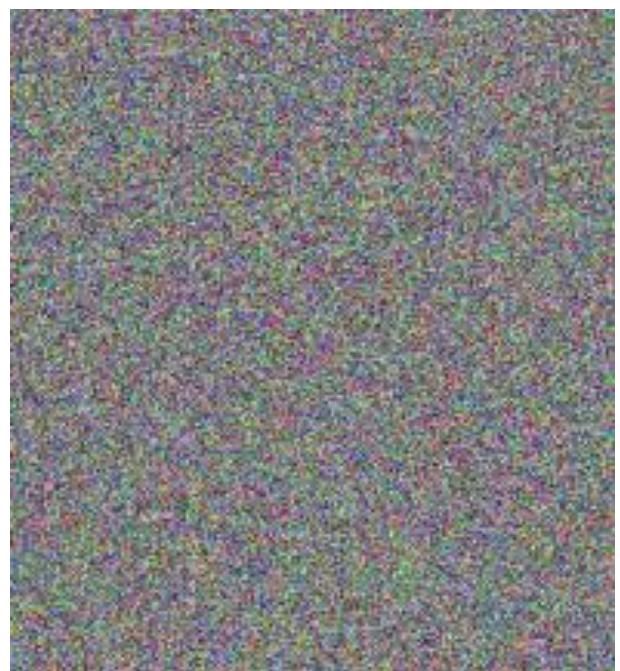
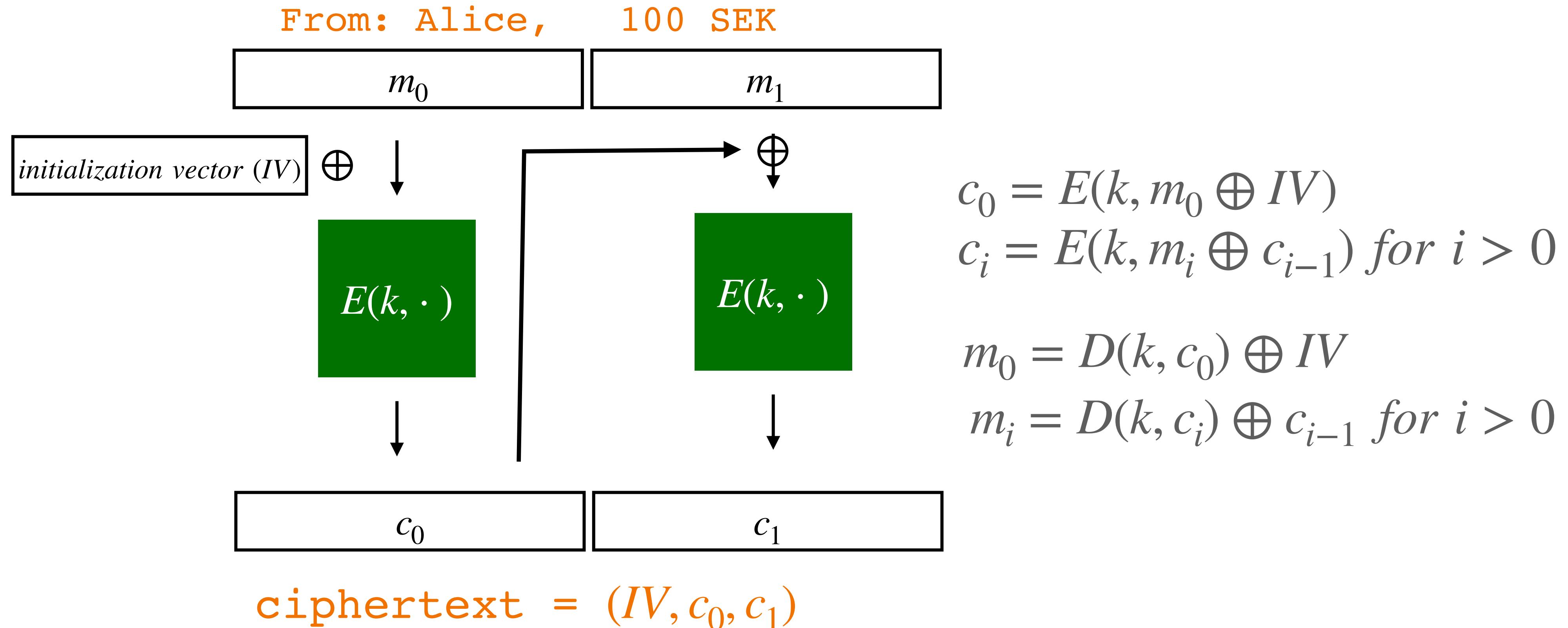
Fact2: Files are often **encrypted** in transit, so this information is not readable to the eavesdropping adversary.



This attack is trivial against AES (or any block cipher) in **CBC mode**

The adversary that launches this attack will succeed with 100% probability AND without knowing the secret key

Cipher Block Chaining Mode (CBC)



The Attack: $IV^* = IV \oplus \text{From : Alice} \oplus \text{From : Eve}$ ciphertext* = (IV^*, c_0, c_1)

⚠️ Encryption alone cannot detect the change, but Bob could. Can you see how?

Integrity Matters. But Even More So Does Authenticating the Source of a Message





**Encryption is not enough!
We need a new cryptographic primitive**

Think Halloween

Lecture 4 - Agenda

Secure Communication

- Towards IND-CPA Security
- Why Does Integrity Matter?

Message Authentication Codes

- Definition + Unforgeability
- CBC-MAC attack
- MAC vs Hash Functions
- HMAC

Authenticated Encryption

- Encrypt-And-MAC
- Encrypt-Then-MAC
- Galois Counter Mode (GCM)

Overview of Module 1

Exercises From Exam

Message Authentication Code (MAC)

Definition: MAC

*Key generation is always trivial: pick
a random key from the key space*

A Message Authentication Code (MAC in short) is a pair of efficient algorithms (MAC, Ver) with the following syntax:

- $MAC : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{T}$ is a probabilistic algorithm that takes in input a key k , a message m and outputs a tag t .
- $Ver : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \rightarrow \{0,1\}$ is a deterministic algorithm that takes in input a key k , a message m and a tag t , and returns 1 (accept) or 0 (reject).

And satisfying the **correctness** condition:

$$Pr[Ver(k, m, MAC(k, m)) = 1] = 1 \text{ for all } k \in \mathcal{K}, m \in \mathcal{M}$$

Protecting Communications Over an Insecure Channel

Goals:

Encryption = prevent any third party from **understanding** the content of the communication

MAC = prevent any third party (or the channel) from **altering** the communication



A tag t is **valid** for a message m against the key k , if $Ver(k, m, t) = 1$

$$b = 1 \text{ if } m^* = m \text{ and } t = t^*$$

$$b = 0 \text{ if } m^* \neq m$$

Aim: quantify the \mathcal{A} 's likelihood in forging a valid tag t^* for a **new** (different) message m^*



What about replay attacks?

Towards a Security Definition

A

Adversary's Goal

~~To decrypt the communication~~

Here we do not care about secrecy, only about integrity

~~To recover the secret key~~

Too strong requirement, damage can be done with less

~~To modify the content of the communication~~

Vague, everyone can “flip bits”

To produce a tag for a *known message* that the receiver will deem **authentic** and that is *different* from what has been sent during the communication

In crypto jargon: **Unforgeability under Chosen Message Attack**



Towards a Security Definition



passive
adversary

Adversary's Goal

To produce a tag that certifies the **authenticity** of a *known* message that is different from what has been sent during the communication

In crypto jargon: **Unforgeability under chosen message attack**

Adversary's Power

Efficient algorithm (probabilistic, and runs in polynomial time $< 2^{60}$)

\mathcal{A} can see everything transmitted over the communication channel

\mathcal{A} knows all details of the MAC scheme except for the secret key
(*Kerckhoffs' principle*)

active
adversary

\mathcal{A} can **drop**, **replace** and **inject** information into the communication channel

Towards a Security Definition

\mathcal{A}

Adversary's Goal

To produce a tag that certifies the **authenticity** of a *known* message that is different from what has been sent during the communication

In crypto jargon: **Unforgeability under chosen message attack**

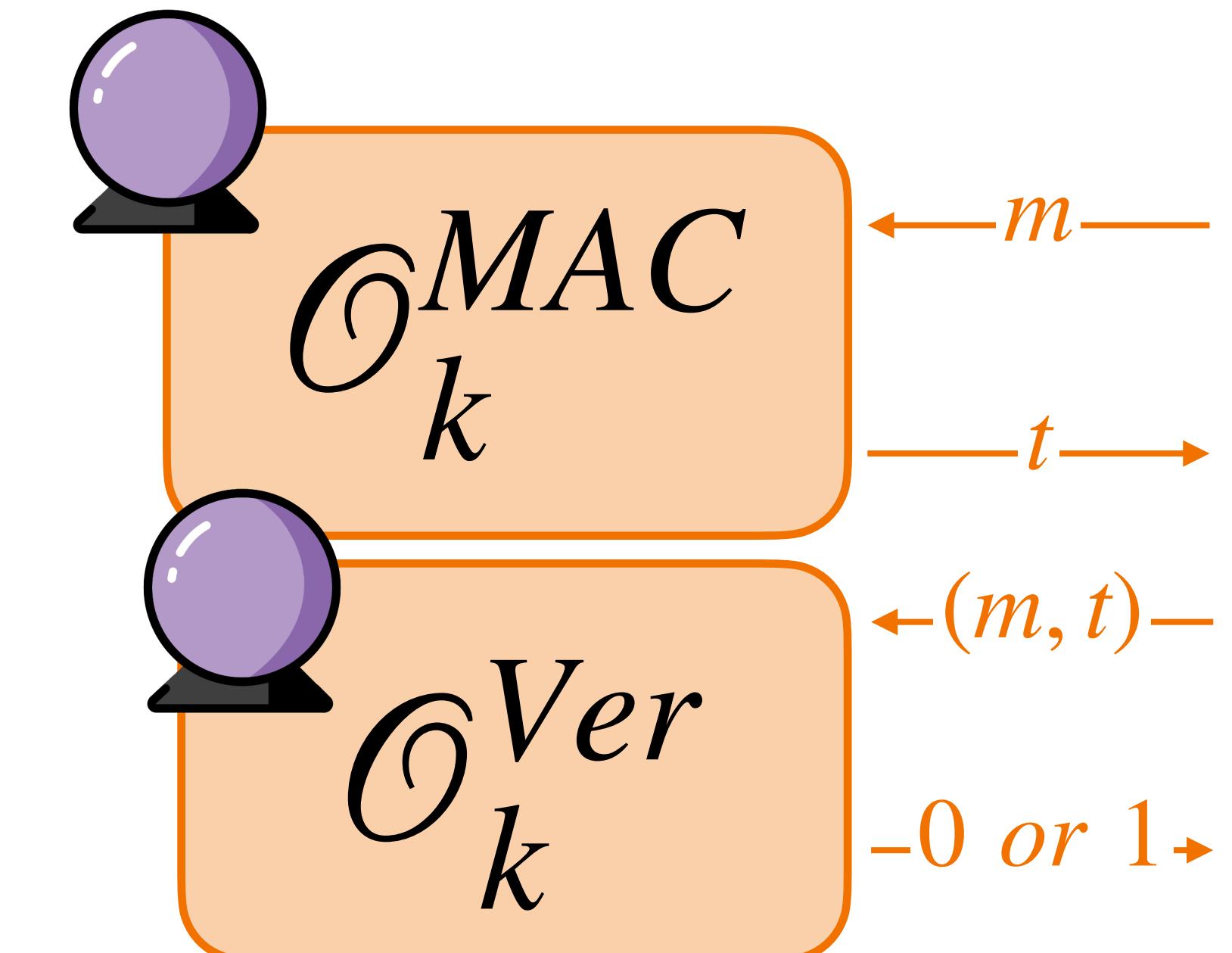
Adversary's Power

\mathcal{A} can **drop**, **replace** and **inject** information into the communication channel
(active adversary)

Adversary's Resources

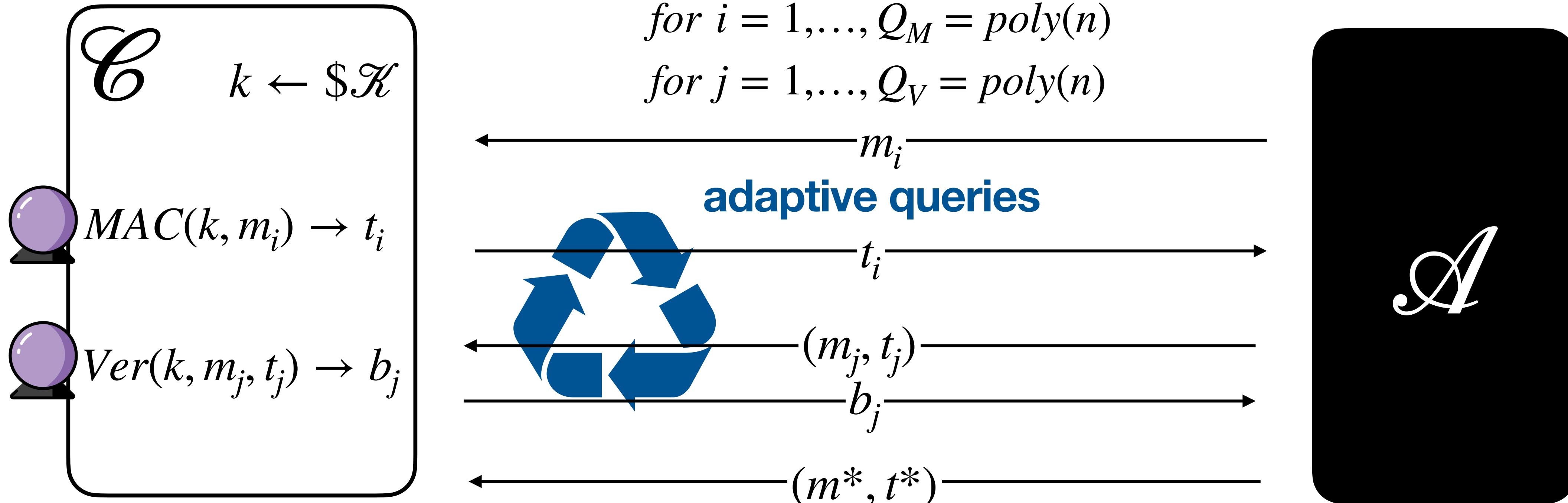
Access to the communication channel

“Oracle Access” to the *keyed* functions $MAC(k, \cdot)$ and $Ver(k, \cdot)$



Security for MACs

Aim: quantify the \mathcal{A} 's likelihood in forging a valid tag t^* for a **new** (different) message m^*



\mathcal{A} wins the security game iff:

$$Ver(k, m^*, t^*) = 1 \text{ AND } m^* \notin \{m_1, \dots, m_{Q_M}\}$$

This security game is called: **Unforgeability under Chosen Message Attack**

Secure MAC

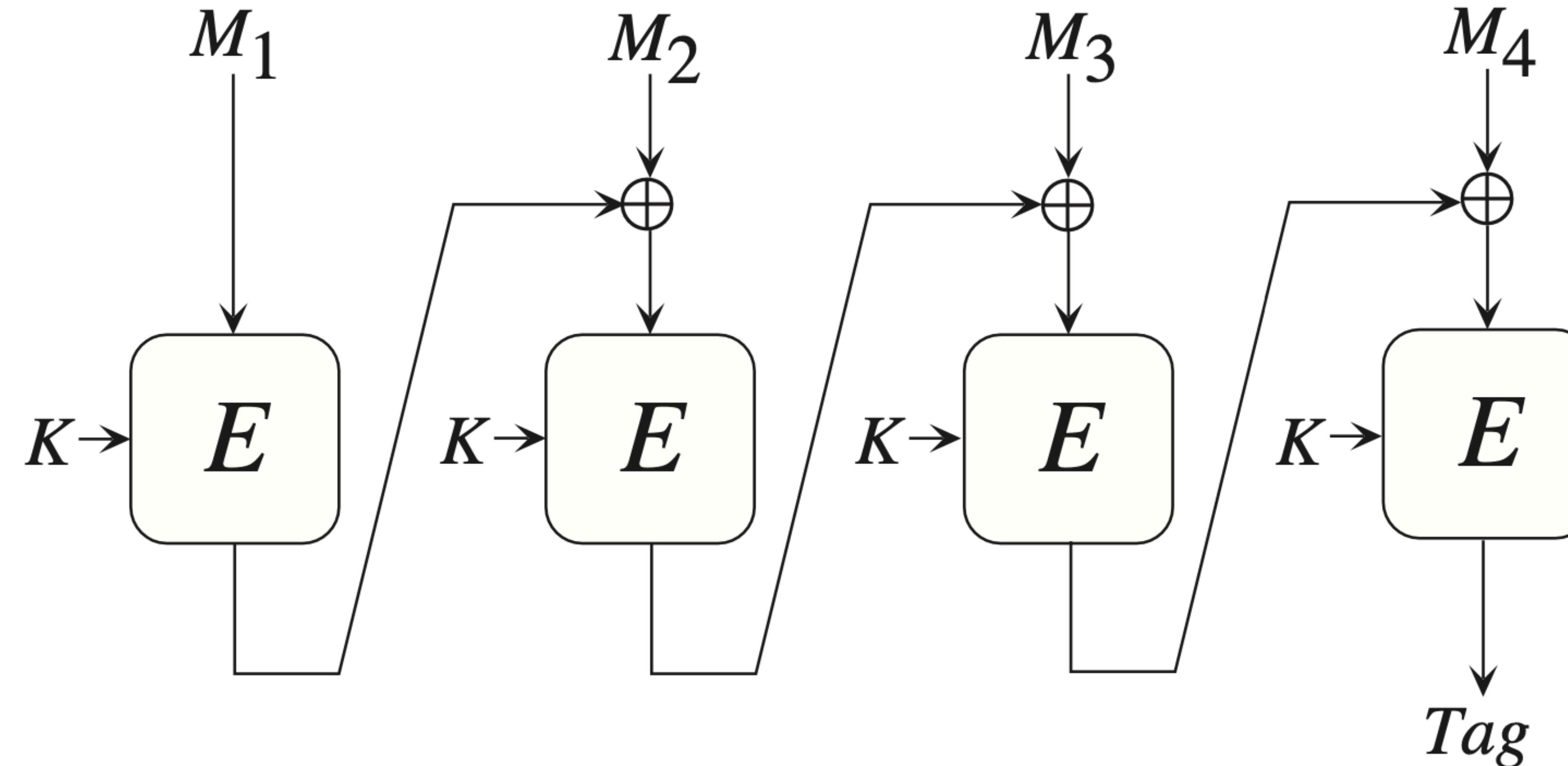
A Message Authentication Code is said to be **secure** (unforgeable under chosen message attack) if **for all efficient** adversaries the probability that \mathcal{A} **wins** the security game is **negligible**. Formally,

$$\Pr[Ver(k, m^*, t^*) = 1 \mid (m^*, t^*) \leftarrow \mathcal{A}^{\mathcal{O}_k^{MAC}, \mathcal{O}_k^{Ver}} \wedge m^* \notin \{m_i\}_{i=1}^{Q_M}] \leq negl(n)$$

In this case n is the size of
the key space $\mathcal{K} = \{0,1\}^n$

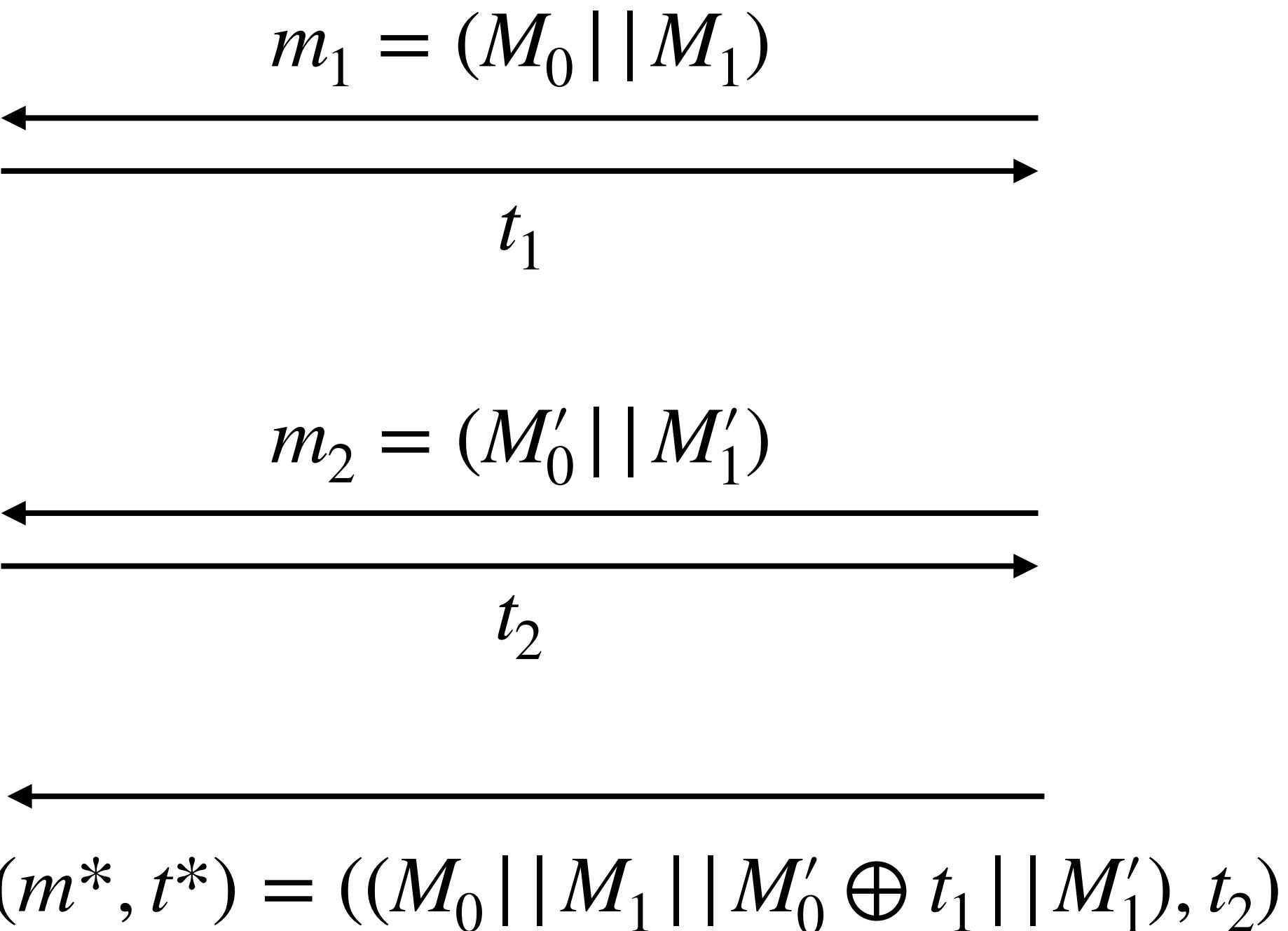
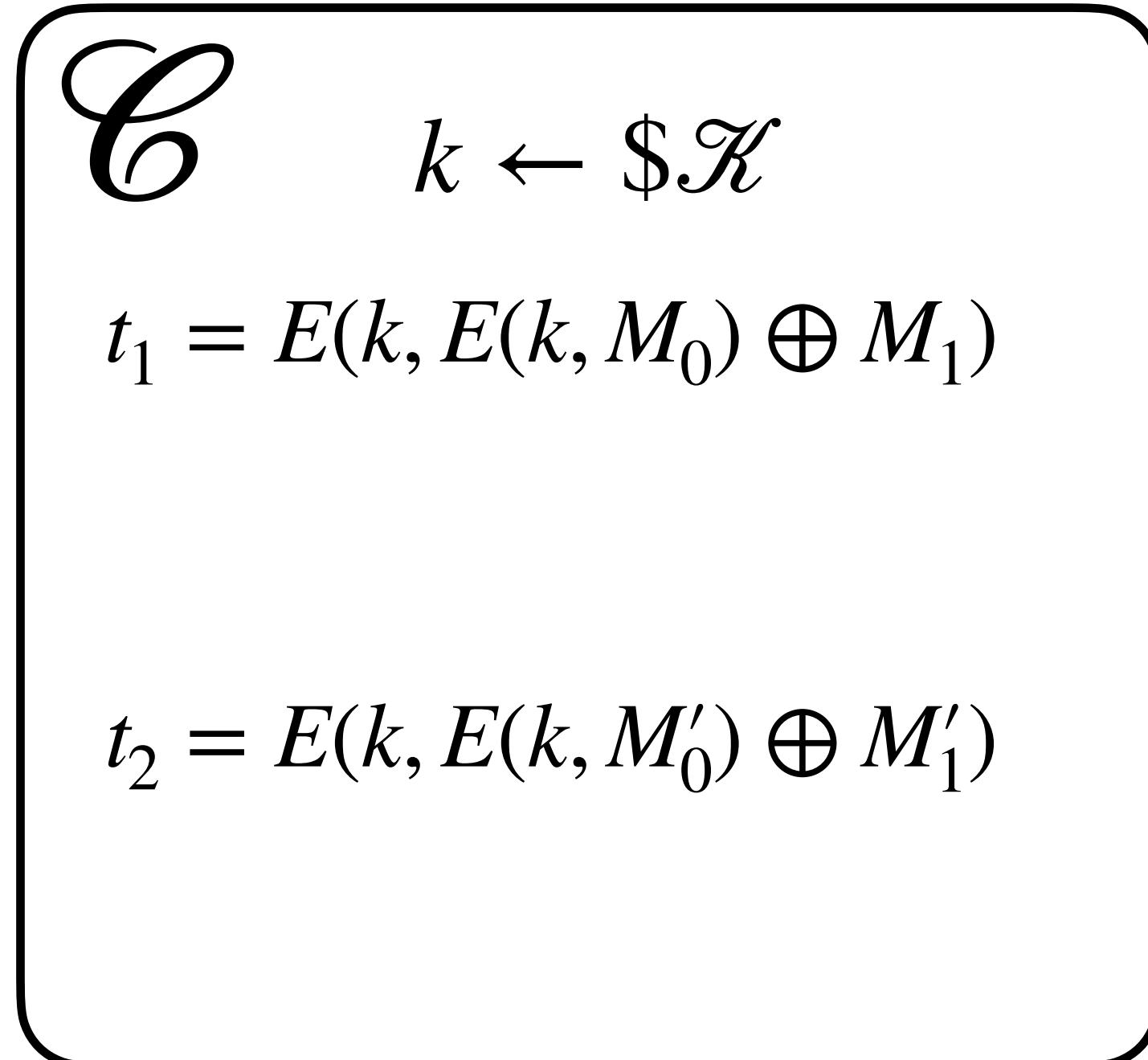
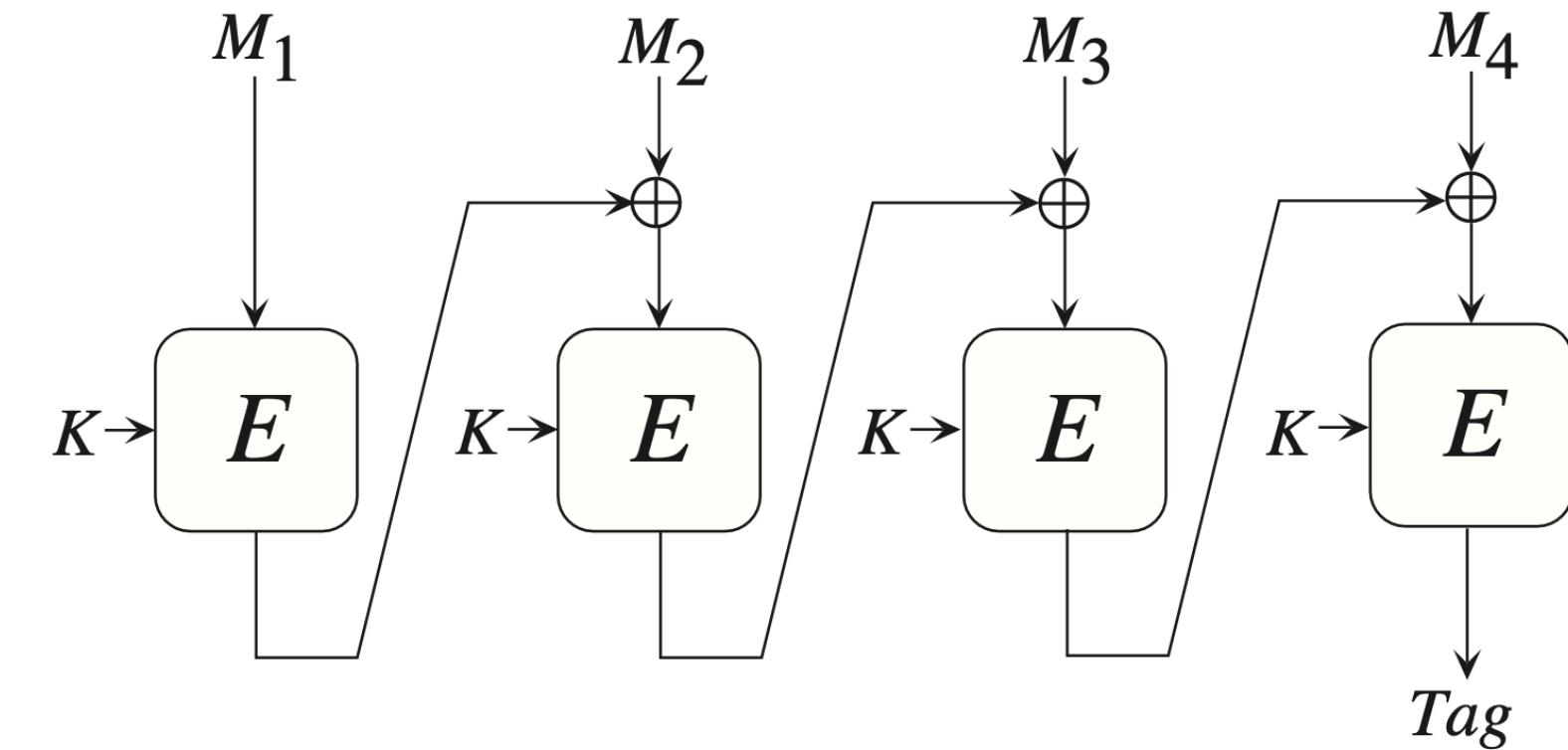
CBC-MAC

⚠️ This RAW version of CBC-MAC is NOT unforgeable. Can you see why?



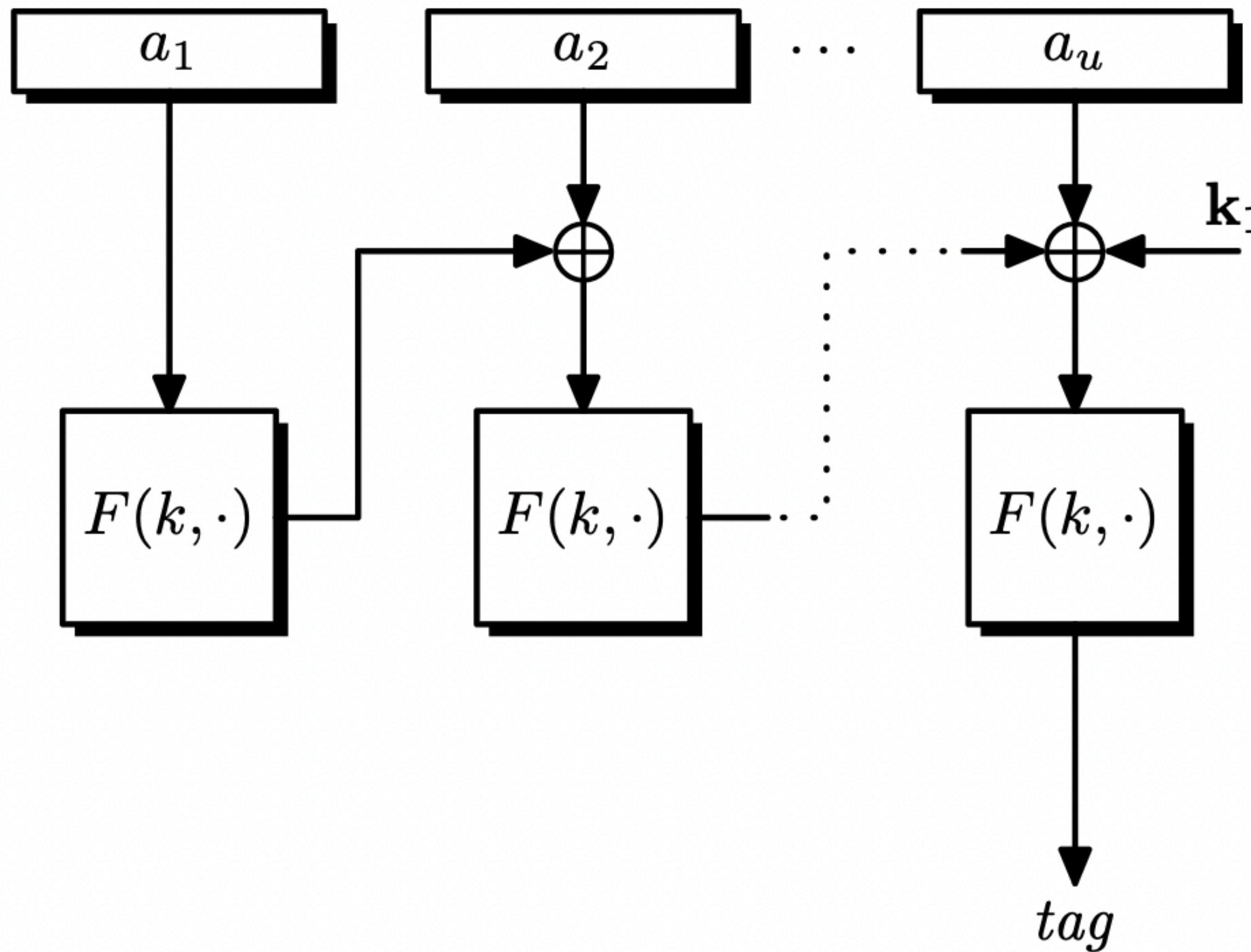
The random IV in CBC encryption mode serves to prevent a dictionary attack on the first ciphertext block. Confidentiality is not a concern for MACs, so $IV=0$ is good enough. The ‘Tag’ is only one block long (so usually shorter than a message, that can be multiple blocks long... + padding)

Raw CBC-MAC Message Extension Attack

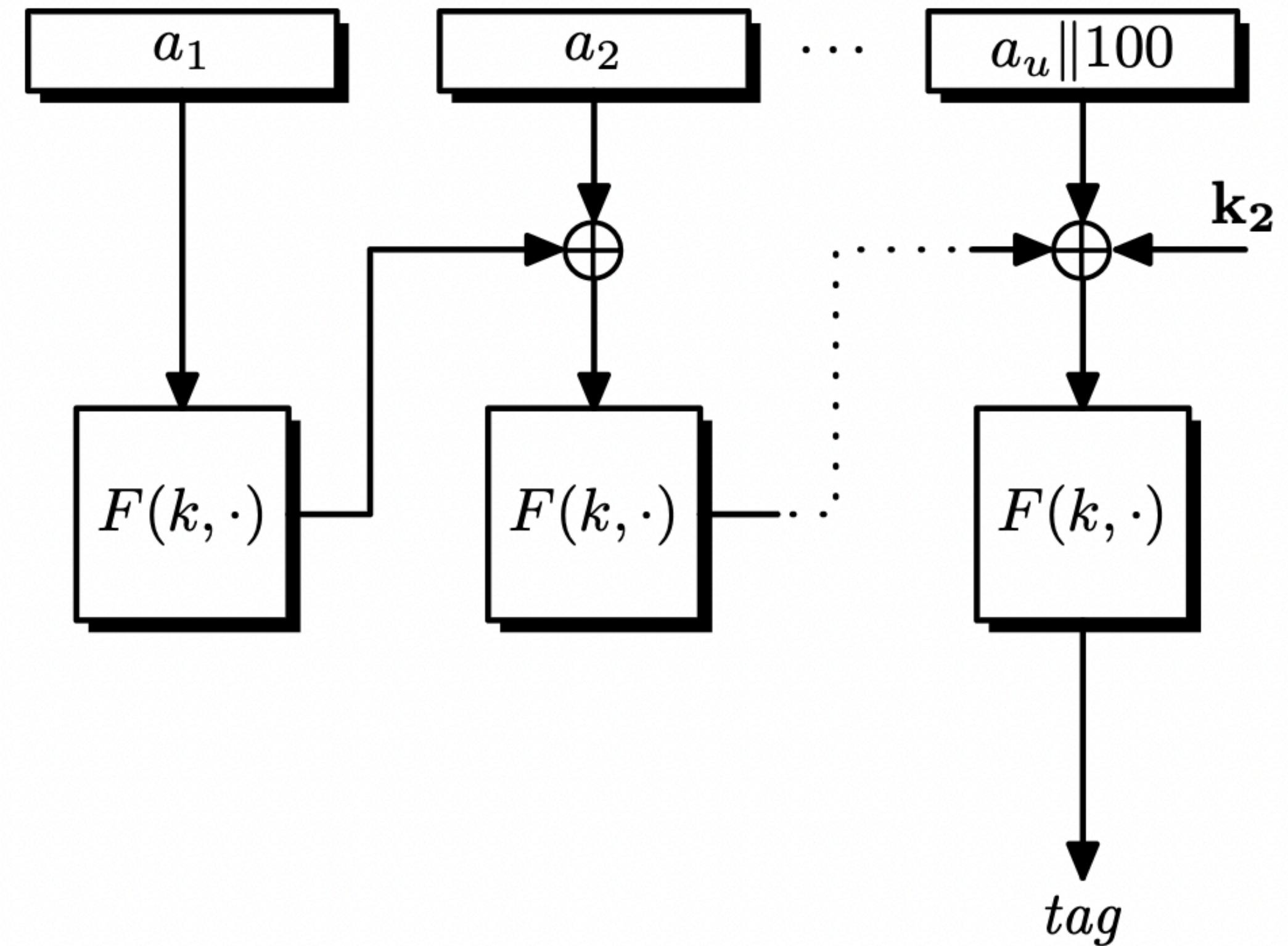


Mitigating the Attack: ANSI CBC-MAC

(a) when $\text{length}(m)$ is a positive multiple of n



(b) otherwise

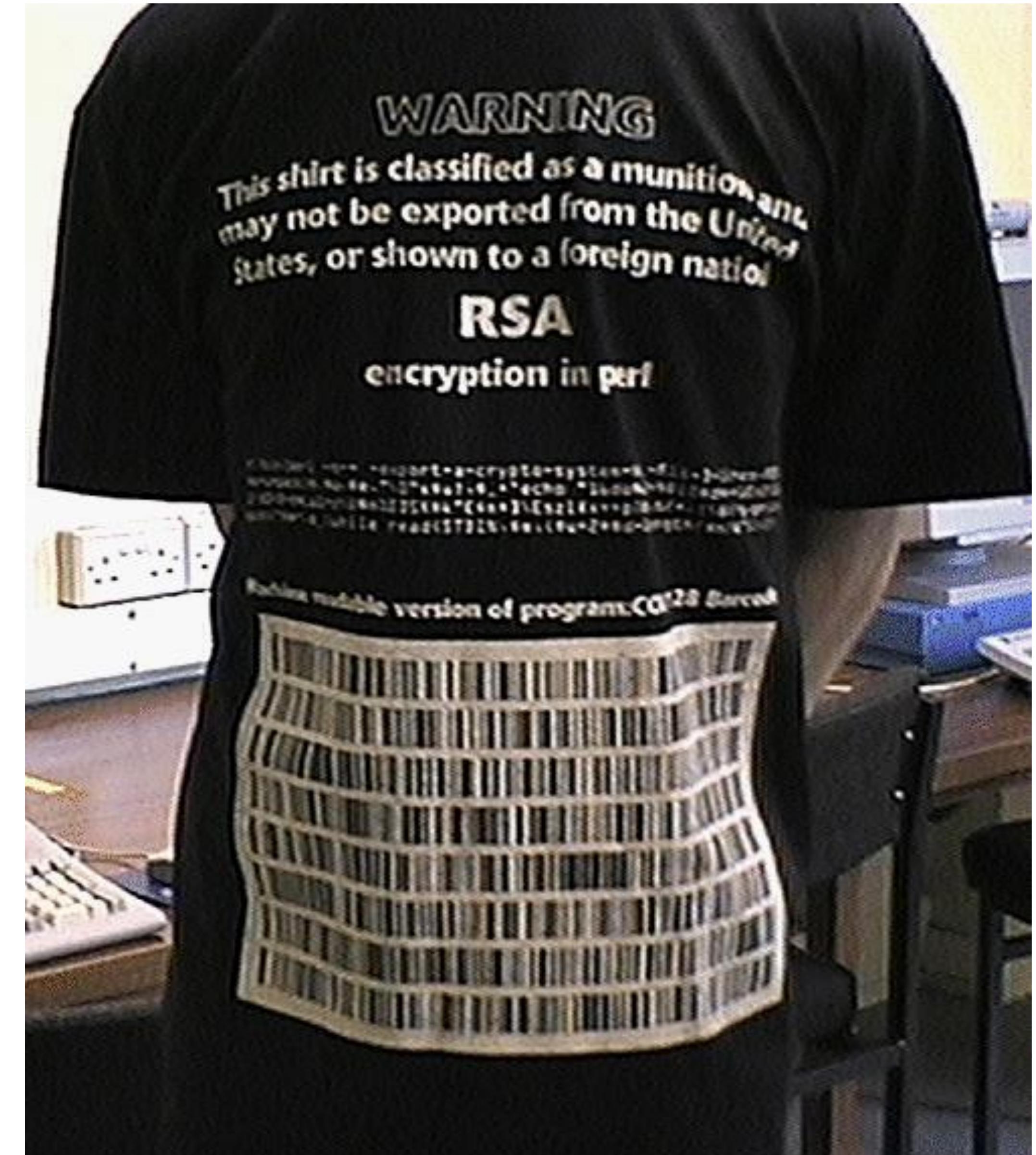


MAC From Block Ciphers VS Hash Functions

- Cryptographic hash functions are usually faster to compute than block ciphers, in software implementations
- The code that implements many hash functions is free, ready to use and can “cross borders” [USA used to restrict the export of cryptographic technologies and devices until 1992!]

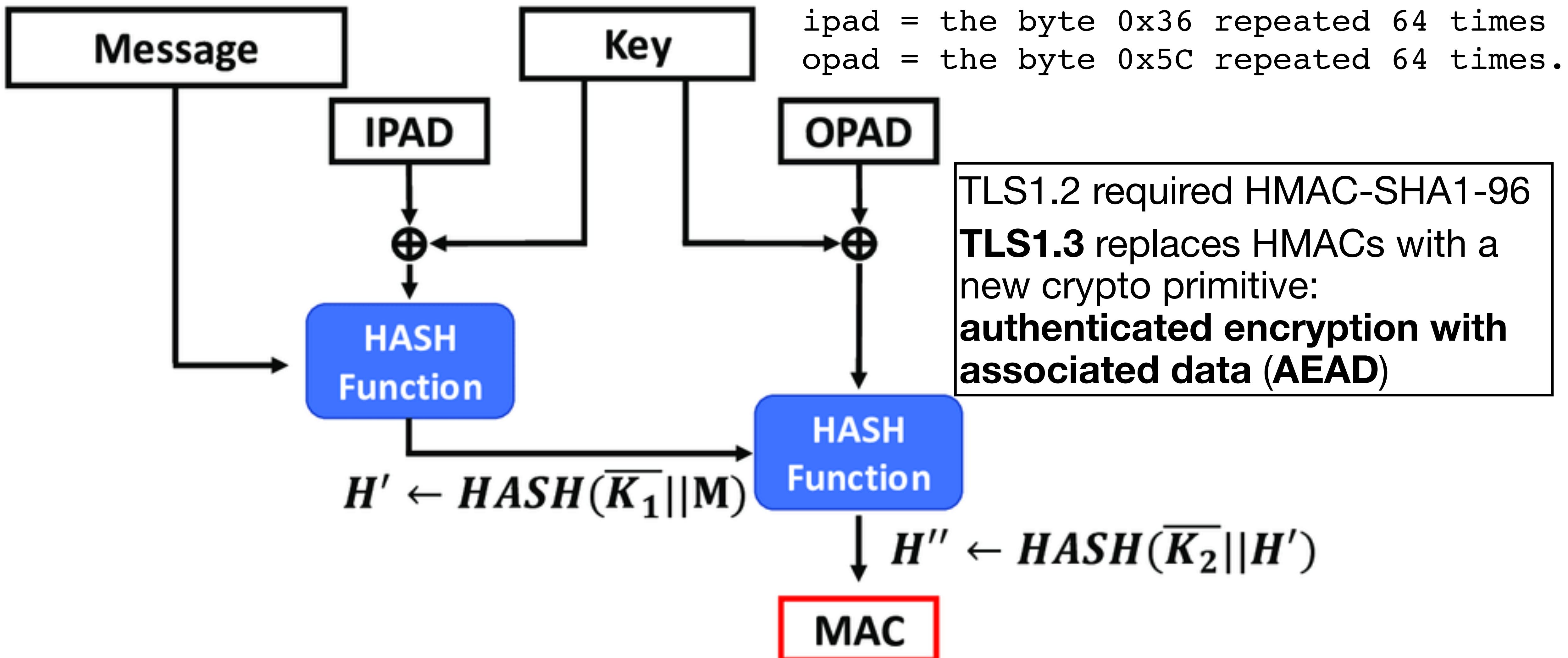
BUT

- Hash functions are not designed for message authentication, and usually do not have keys! How to go about this?
- **HMAC for internet security (TLS1.2, SSH)**



HMAC

$$HMAC(k, text) = H(k \oplus \text{opad} || H(k \oplus \text{ipad} || text))$$



Lecture 4 - Agenda

Secure Communication

- Towards IND-CPA Security
- Why Does Integrity Matter?

Message Authentication Codes

- Definition + Unforgeability
- CBC-MAC attack
- MAC vs Hash Functions
- HMAC

Authenticated Encryption

- Encrypt-And-MAC
- Encrypt-Then-MAC
- Galois Counter Mode (GCM)

Overview of Module 1

Exercises From Exam

Authenticated Encryption (With Associated Data)



CONFIDENTIALITY
&
INTEGRITY
at the same time

Authenticated Encryption via Generic Composition

- c_1 may leak information about m
- decryption happens before the integrity check



what can go bad?

Encrypt-and-MAC:

AE.Encrypt

```
Split  $k = (k_0 \parallel k_1)$ 
 $c_0 \leftarrow E(k_0, m)$ 
 $c_1 \leftarrow MAC(k_1, m)$ 
return  $c = (c_0, c_1)$ 
```

AE.Decrypt

```
Split  $k = (k_0 \parallel k_1)$ 
 $m \leftarrow D(k_0, c_0)$ 
 $b \leftarrow Ver(k_1, m, c_1)$ 
if  $b = 1$  return  $m$ 
Else return  $\perp$ 
```

*This is the most secure way to compose the two primitives
It is used in TLS1.3, IPsec, GCM*



Encrypt-then-MAC:

AE.Encrypt

```
Split  $k = (k_0 \parallel k_1)$ 
 $c_0 \leftarrow E(k_0, m)$ 
 $c_1 \leftarrow MAC(k_1, c_0)$ 
return  $c = (c_0, c_1)$ 
```

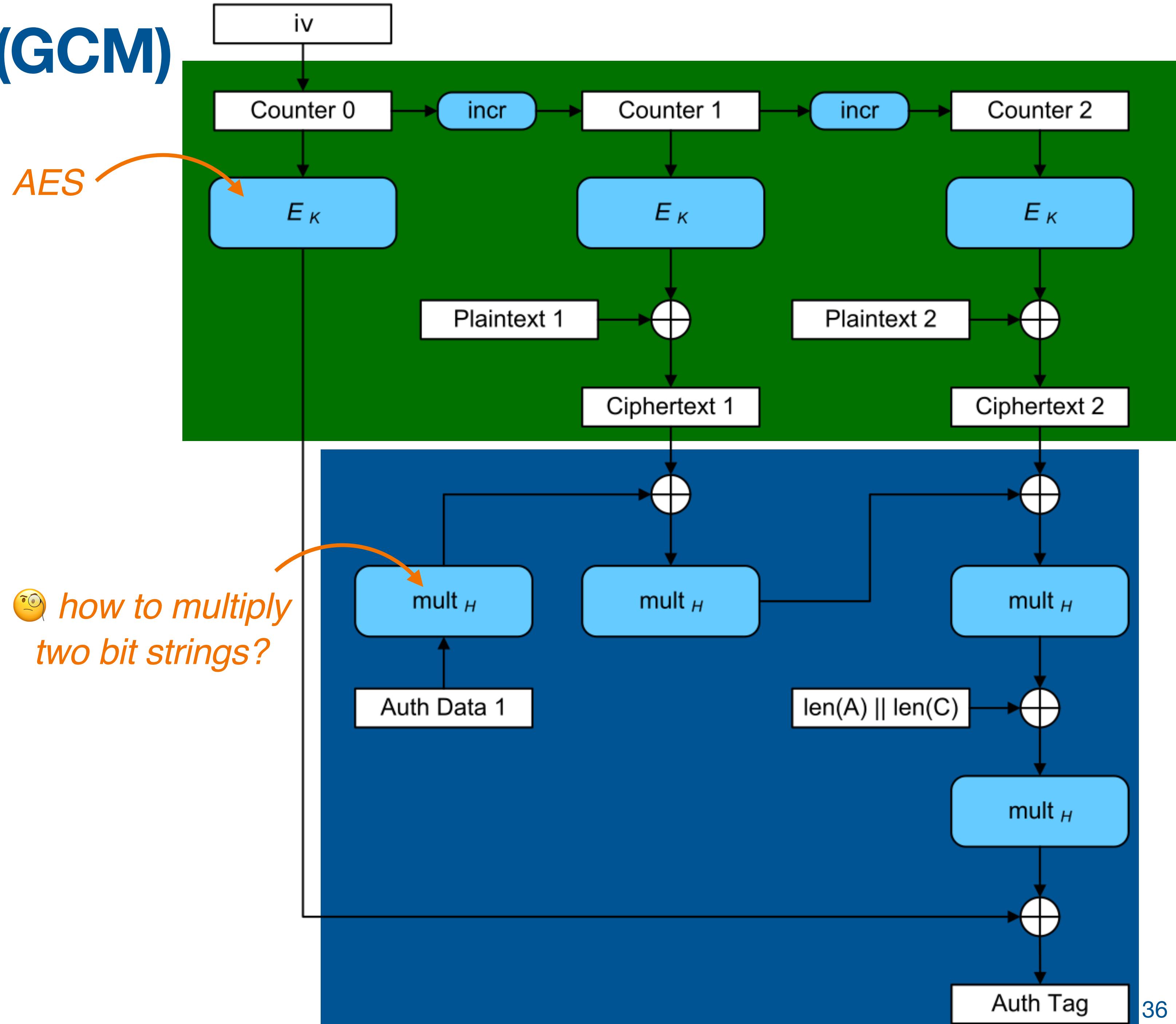
AE.Decrypt

```
Split  $k = (k_0 \parallel k_1)$ 
 $b \leftarrow Ver(k_1, c_0, c_1)$ 
if  $b = 1$  return  $D(k_0, c_0)$ 
Else return  $\perp$ 
```

There are many ways to combine a cipher and a MAC, not all combinations are secure!

Galois Counter Mode (GCM)

- GCM is a construction of **Encrypt-then-MAC** style of authenticate encryption
- GCM is widely adopted for its **great performance**
- GCM provides data authenticity **(integrity) and confidentiality simultaneously**
- GCM may also authenticate plaintext **Associated Data (AEAD)**, e.g., headers



Galois Field Multiplication

[not in exam]

Intuition:

- 1- see bit-strings as vectors with coefficients over \mathbb{Z}_2
- 2- see vectors as polynomials
- 3- we know how to multiply polynomials

Math caveat

In order to make sure the result of the multiplication is *always* a bit-string of length 128 we need to do operations in a special mathematical object called **Galois Field**

$$GF(2^{128}) := \mathbb{Z}_2[x]/(x^{128} + x^7 + x^2 + x + 1)$$

If you want to read up on this (optional):

<https://eprint.iacr.org/2004/193.pdf>

<https://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf>

Lecture 4 - Agenda

Secure Communication

- Towards IND-CPA Security
- Why Does Integrity Matter?

Message Authentication Codes

- Definition + Unforgeability
- CBC-MAC attack
- MAC vs Hash Functions
- HMAC

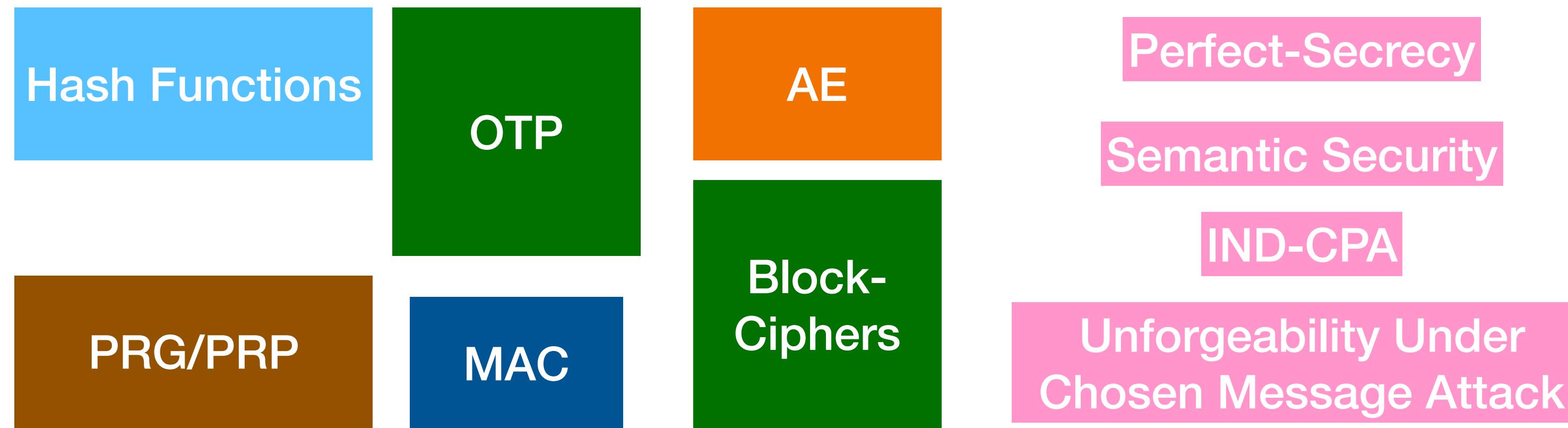
Authenticated Encryption

- Encrypt-And-MAC
- Encrypt-Then-MAC
- Galois Counter Mode (GCM)

Overview of Module 1

Exercises From Exam

Overview of Module 1



Now you can understand ~70% of the cryptographic tools used nowadays

What's left?

- Public key encryption
- Key exchange protocols
- Digital signatures and Certificates
- Proof Systems (NIZK, SNARK)
- MPC (secure multi party computation)
- Privacy Enhancing Technologies

Please Give Us Your Feedback on How You're Liking Team Teaching



<https://forms.office.com/e/BeR1L5hE7R>

Lecture 4 - Agenda

Secure Communication

- Towards IND-CPA Security
- Why Does Integrity Matter?

Message Authentication Codes

- Definition + Unforgeability
- CBC-MAC attack
- MAC vs Hash Functions
- HMAC

Authenticated Encryption

- Encrypt-And-MAC
- Encrypt-Then-MAC
- Galois Counter Mode (GCM)

Overview of Module 1

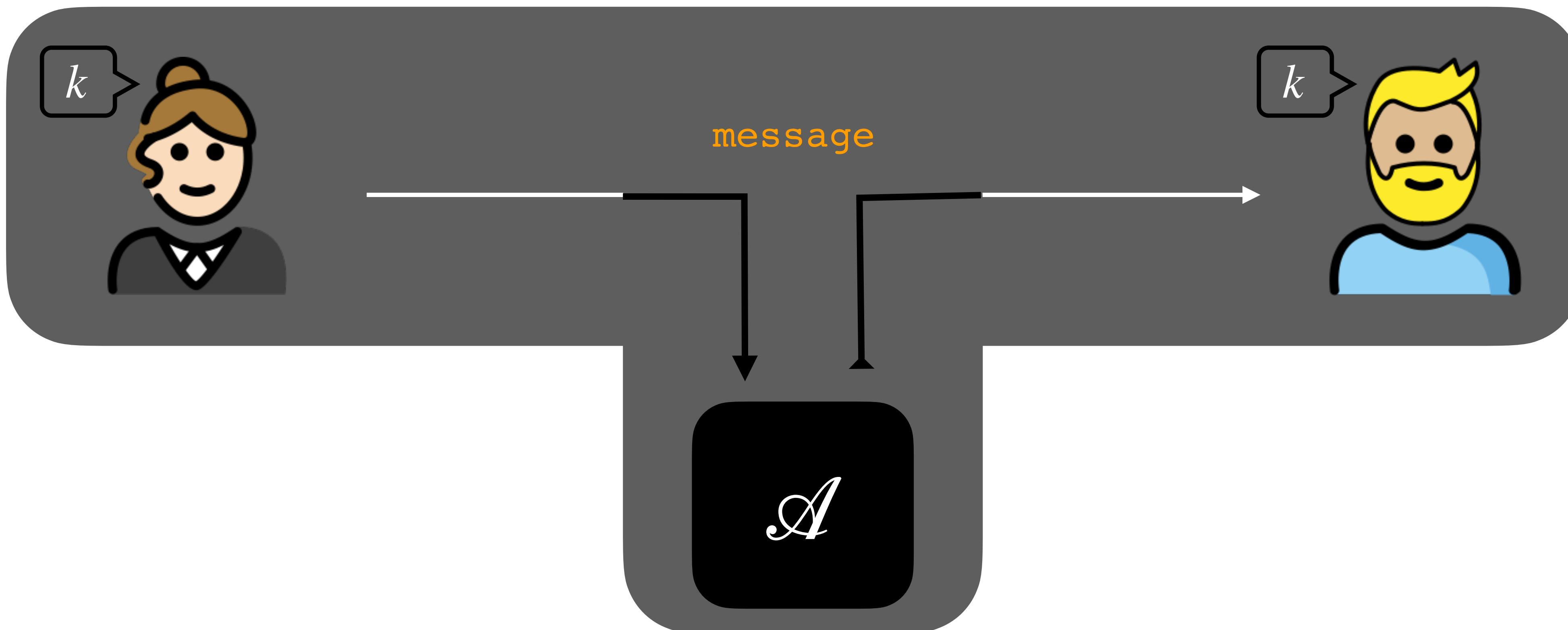
Exercises From Exam

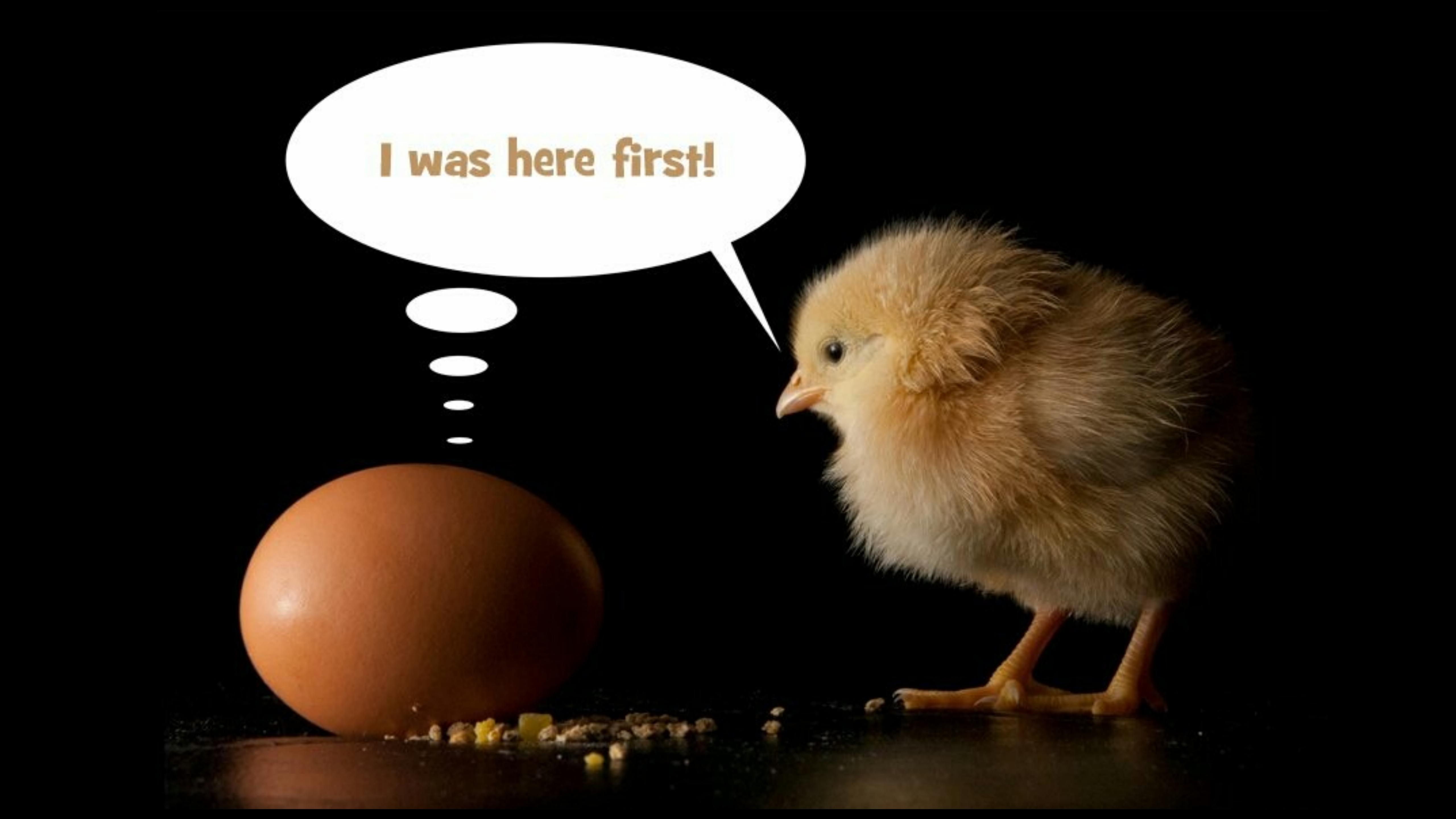
HA1 First Hand in Deadline Is TODAY

If you have doubts, use the **Office Hours 11:30-12:30 on Monday(s) in EDIT 6128**.
The PhD TA responsible for grading HAs will be there happy to answer your questions!
Do not ask me/the lecturer about tasks in HAs.

...Back in Module 1...

- Perfectly secure encryption (**OTP**)
- Semantically secure encryption (**PRG**)
- IND-CPA secure encryption (**AES, Block Ciphers**)
- Hash Functions, Merkle Trees, Blockchains...
- Integrity (**MAC, AEAD**)





I was here first!



Lecture Agenda

Key-Exchange

- Problem Statement
- A Simple Solution
- Formalisation: **Group Theory**

Diffie-Hellman Key Exchange (DH)

- The Protocol
- On the Bit Security of DH Keys
- Securing DH Keys
- MiM Attack

Security for DH

- DL, CDH, DDH Problems
- Relation Between Problems
- $\text{CDH} \leq \text{DL}$ (**Poof**)

Public Key Cryptography

- Introduction
- One-Way Trapdoor Functions
- The Hash-and-Sign Paradigm

Problem Statement

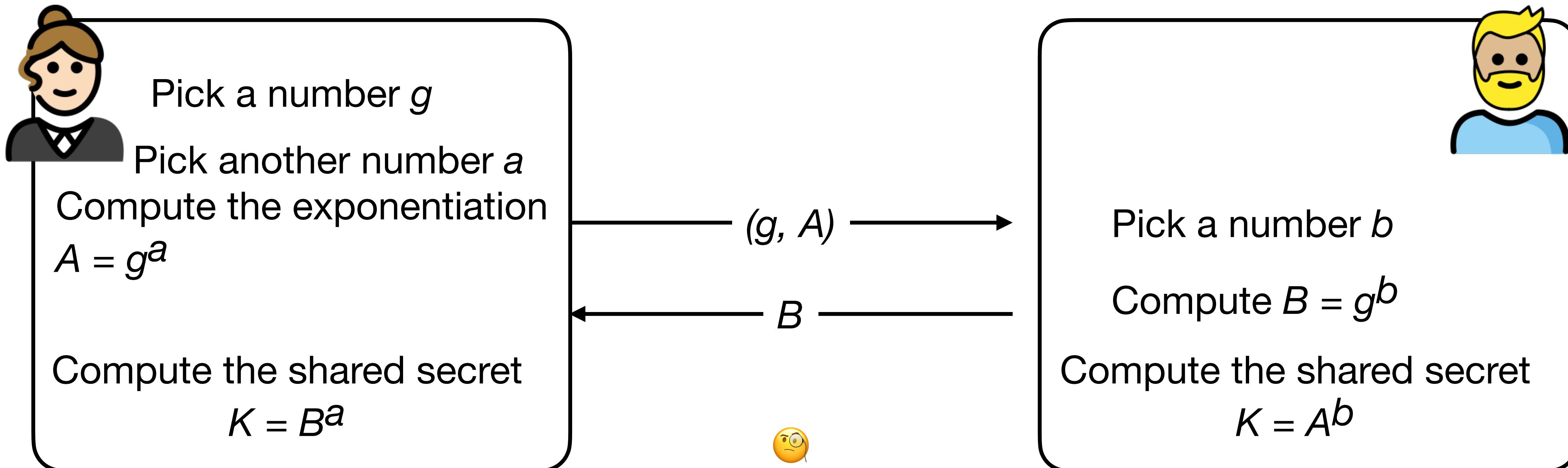
Alice and Bob want to find a way to share a secret key k without relying on a previously shared secret **AND** they want to do so, using a public channel, that is monitored* by the Adversary



Tool: Exponentiation g^x

*For now we only consider passive \mathcal{A} (eavesdropper)

A Simple Protocol



Why is K the same for both?

 *What prevents A from learning K given (g, A, B) ?*

In theory (=without bounds on the computational power of the adversary) nothing!

In practice, this challenge is (computationally) hard to solve, **if** you work in the correct domain



Let's get formal!

AI generated picture from easy-peasy.ai

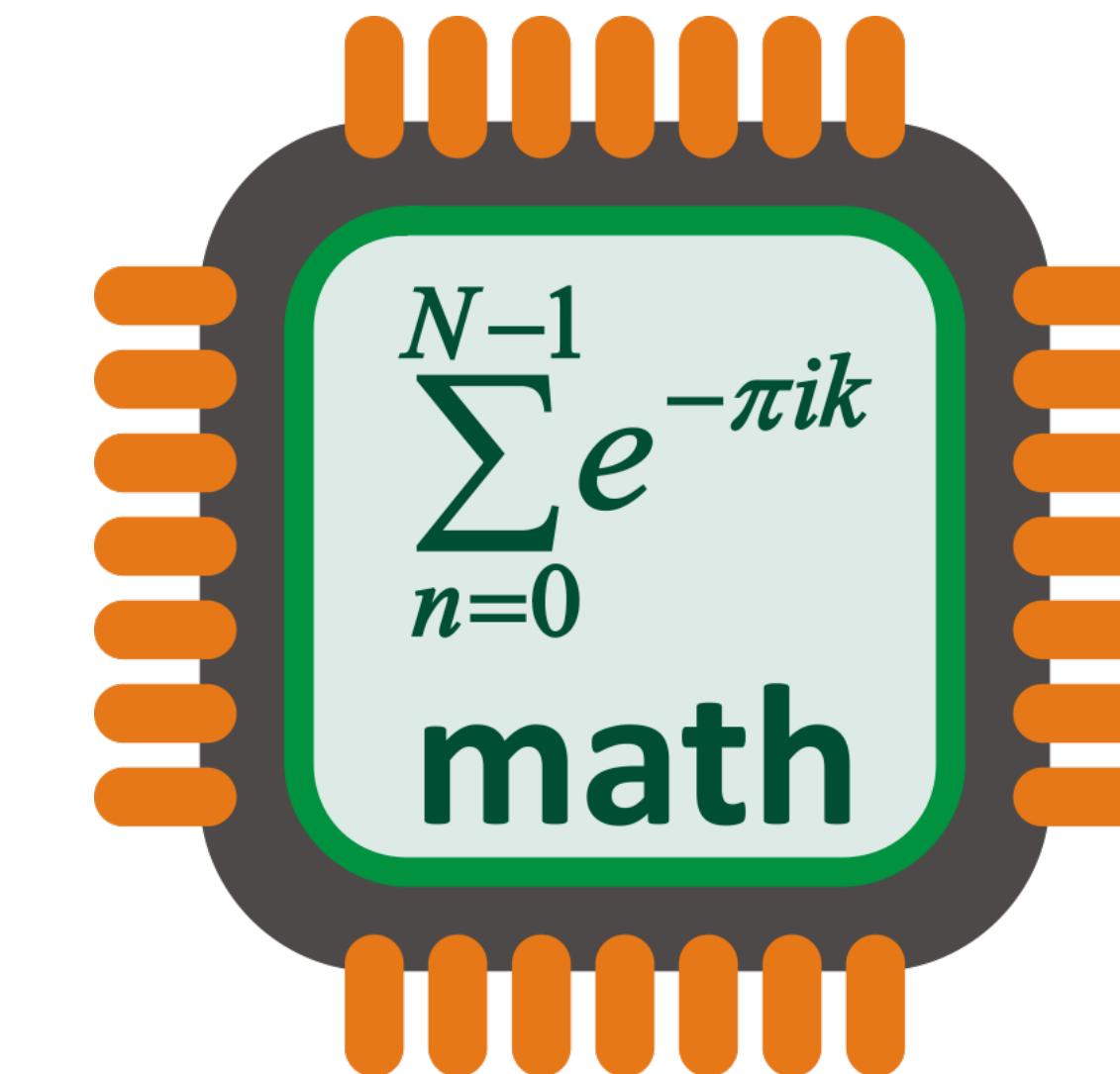
A First Attempt

g is a prime number and the exponents, a,b , are large positive integers

$$30226801971775055948247051683954096612865741943 = 7^?$$

This approach could work, but there is no upper limit on **how large** A,B,K may become.

Moreover, if we want to use K for a symmetric encryption scheme, K needs to be encoded into an n -bit string, for some **fixed** value n .

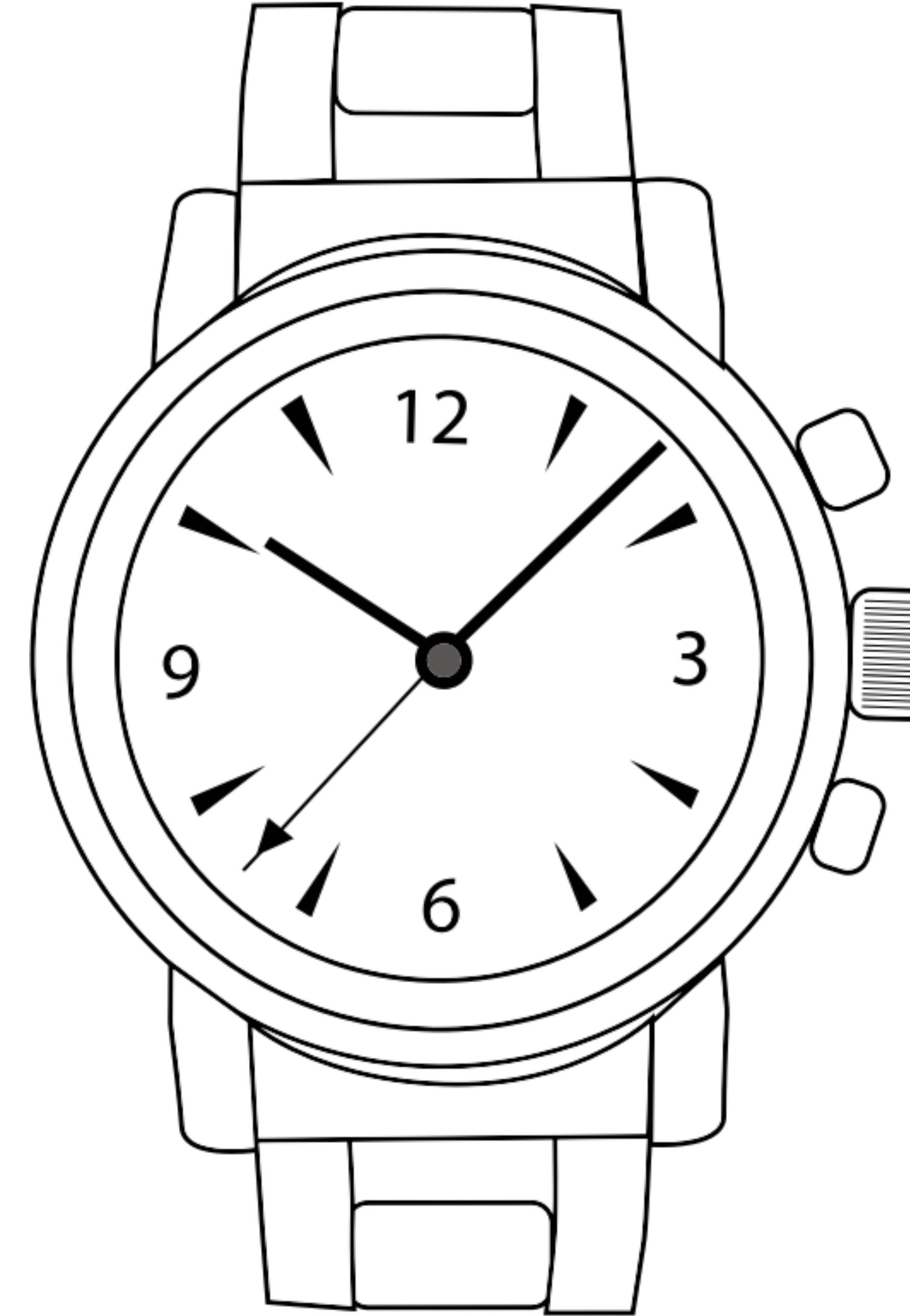


WANTED: a mathematical object that allows us to do arbitrary exponentiations while guaranteeing the values we get stay within a certain range.

Groups That You Already Know

\mathbb{Z}_{12}

$$3 + 5 = ? \pmod{7}$$



\pmod{n}

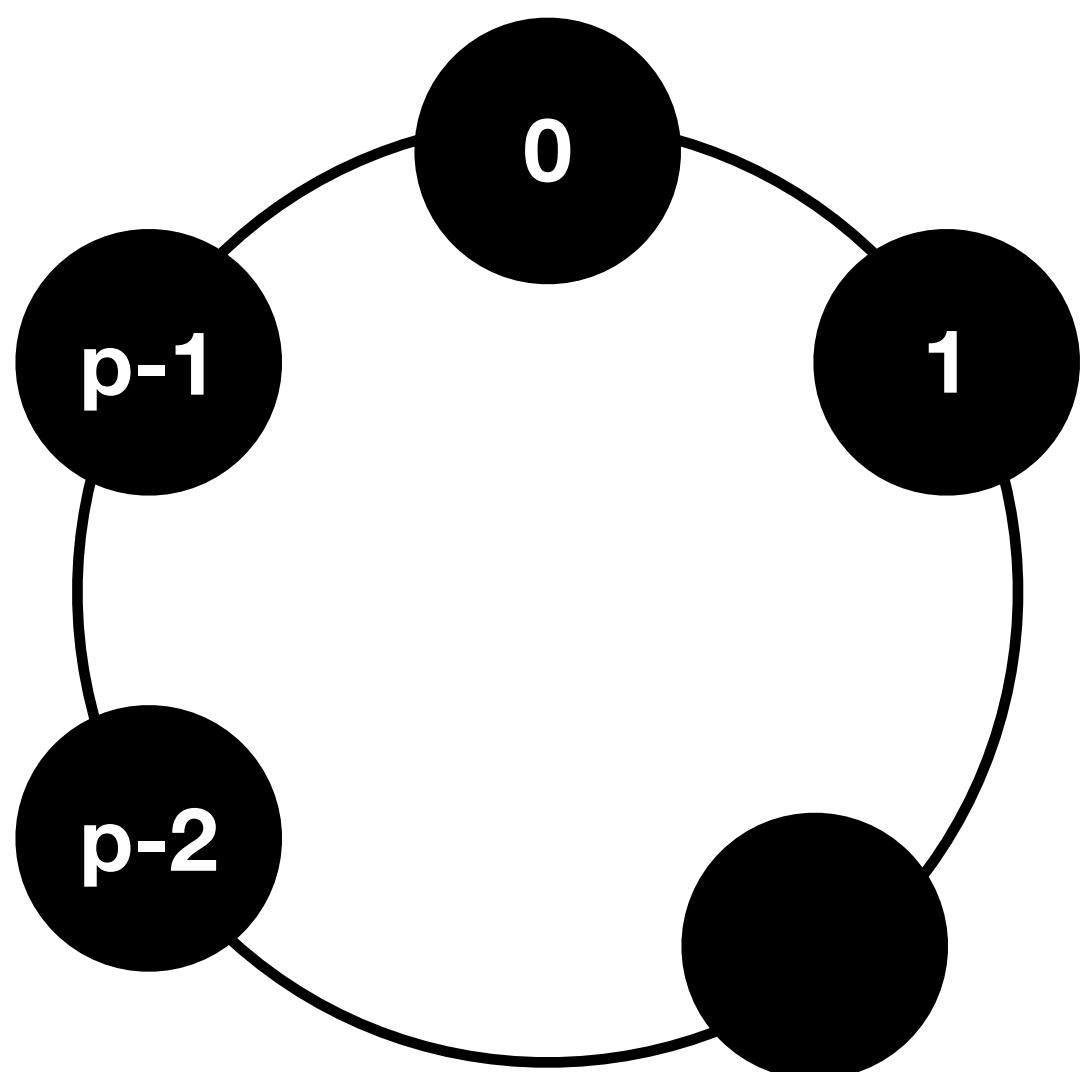
$$3^5 = ? \pmod{7}$$

Group

Think $\mathbb{G} = (\mathbb{Z}_p, +)$

$$\mathbb{Z}_p = \{0, 1, \dots, p - 1\}$$

$$g \star h = g + h \pmod{p}$$



Definition: Group

A group \mathbb{G} is a finite set of elements (usually also denoted at \mathbb{G}) together with an operation \star , that is, a function $\star: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}$ with the following properties:

- Closure:** for all $g, h \in \mathbb{G}$ it holds that $g \star h \in \mathbb{G}$
- Associativity:** for all $g, h, k \in \mathbb{G}$ it holds that
$$(g \star h) \star k = g \star (h \star k)$$
- Identity:** There exists an element $e \in \mathbb{G}$ such that
$$e \star g = g \star e = g \text{ for all } g \in \mathbb{G}$$
- Inverse:** for every $g \in \mathbb{G}$ there exists a (unique!) element $\bar{g} \in \mathbb{G}$ such that $g \star \bar{g} = e$.

Important Concepts for Groups

An element $g \in \mathbb{G}$ is called a **generator** for \mathbb{G} if all elements in the group can be seen as ‘multiples’ of g , i.e., $\langle g \rangle = \{g, g \star g, g \star g \star g, \dots\} = \mathbb{G}$.

A group \mathbb{G} that admits a **generator** $g \in \mathbb{G}$ is called **cyclic group**.

A group (\mathbb{G}, \star) is **commutative (abelian)** if for any two elements $g, h \in \mathbb{G}$ it holds that $g \star h = h \star g$.

The **order** of a group is number of elements in it (its cardinality), i.e., $ord(\mathbb{G}) = |\mathbb{G}|$. The order of an element $h \in \mathbb{G}$ is the smallest positive integer $d = ord(h)$ such that $\underbrace{h \star h \star \dots \star h}_{d \text{ times}} = e$ the identity element in \mathbb{G} .

Topics useful for Tasks 1,2,3 and most Questions in Task 6, in HA2

Facts

- All cyclic groups are commutative
- The generator is not unique (if a group admits a generator, then it admits multiple generators)
- If a group has order n then for all $g \in \mathbb{G}$: $\underbrace{g \star g \dots \star g}_{n \text{ times}} = e$
- For every $h \in \mathbb{G}$ it holds that $ord(h)$ divides $ord(\mathbb{G})$.

Results on Group Theory

Exercise Let $N > 1$ be an integer, then $(\mathbb{Z}_N, +)$ is a group.

Exercise (\mathbb{Z}_N, \cdot) is not a group.

Exercise Let $N > 1$ be an integer, then (\mathbb{Z}_N^*, \cdot) is a group.

$$\mathbb{Z}_N^* = \{x \in \{1, 2, \dots, N-1\} \mid \gcd(x, N) = 1\}$$

the set of all elements in \mathbb{Z}_N that have a multiplicative inverse (modulo N)



are these two definitions equivalent?

the (extended) Euclidian algorithm computes $s, t \in \mathbb{Z}$ s.t.
 $xs + Nt = \gcd(x, N) = 1$

taking the equality modulo N we get:
 $xs \equiv 1 \pmod{N}$, i.e., $x^{-1} \equiv s \pmod{N}$

Example on the Board

$$\mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$\langle 1 \rangle = 1$$

$$\langle 2 \rangle = \{2^0 = 1, 2^1, 2^2 = 4, 2^3 = 8, 2^4 = 16 \equiv 5 \pmod{11}, 2^5 = 2 \cdot 2 \cdot 2^4 = 2 \cdot 5 = 10 \pmod{11},$$

$$2^6 = 2^5 \cdot 2 = (-1) \cdot 2 = 9 \pmod{11}, 2^7 = 7 \pmod{11}, 2^8 = 3 \pmod{11},$$

$$2^9 = 6 \pmod{11}, 2^{10} = 12 \equiv 1 \pmod{11}\}$$

↑ cycles back to 1

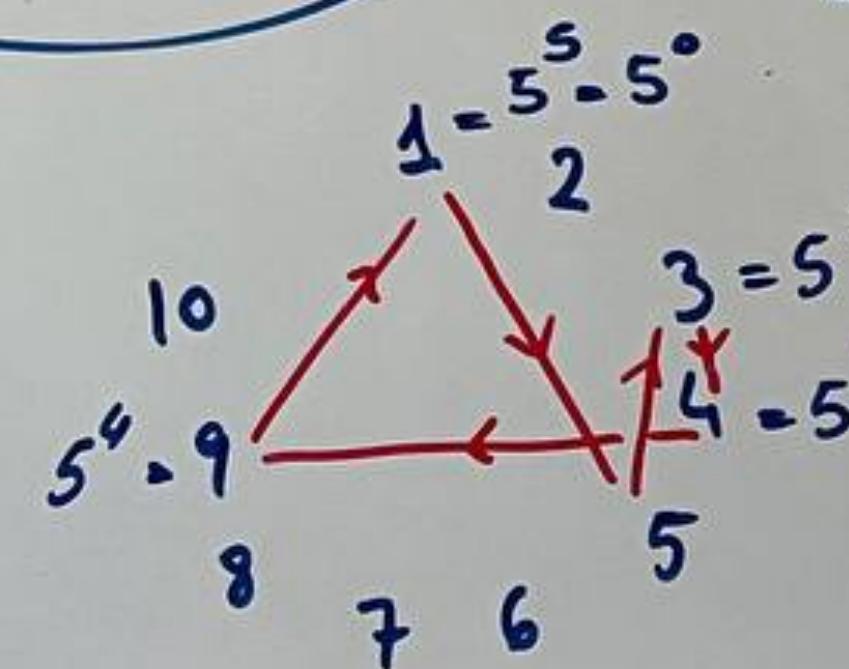
$$|\langle 2 \rangle| = 10 = |\mathbb{Z}_{11}^*|$$

2 is a generator

$$\underline{\langle 5 \rangle} = \{1, 5, 3, 4, 9, 1\}$$

5 is not a generator of \mathbb{Z}_{11}^*

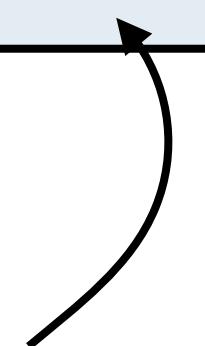
5 generates a subgroup of order 5



Euler's Totient Function

Euler's Totient function $\varphi : \mathbb{N} \rightarrow \mathbb{N}$: given in input a number N , φ counts the number of positive integers smaller than N that are relatively prime to N . This number is computed as:

$$\varphi(N) = N \prod_{p|N} \left(1 - \frac{1}{p}\right)$$



“ p divides N ”
 p is a prime smaller than N

$$|\mathbb{Z}_N^*| = \varphi(N)$$

Useful re-formulations:

$$\varphi(p) = p - 1$$

$$\varphi(N) = (p-1)(q-1) \text{ if } N = pq \text{ and } p, q \text{ are prime numbers}$$

🧐 can you see why?

Important Results on Group Theory

Theorem (Euler) For all $a \in \mathbb{Z}_N^*$ it holds that $a^{\varphi(N)} = 1 \pmod{N}$.

Corollary (Fermat's Little Theorem)

Let p be a prime number, then $\forall a \in \mathbb{Z}_p^*, a^{p-1} = 1 \pmod{p}$

Corollary Let $N > 2$ be a positive integer. For all $a \in \mathbb{Z}_N^*, x \in \mathbb{N}$,
it holds that $a^x \pmod{N} = a^{x \pmod{\varphi(N)}} \pmod{N}$



use these results to compute $7^{32} \pmod{11}$ without a calculator

Lecture Agenda

Key-Exchange

- Problem Statement
- A Simple Solution
- Formalisation: **Group Theory**

Diffie-Hellman Key Exchange (DH)

- The Protocol
- On the Bit Security of DH Keys
- Securing DH Keys
- MiM Attack

Security for DH

- DL, CDH, DDH Problems
- Relation Between Problems
- $\text{CDH} \leq \text{DL}$ (**Poof**)

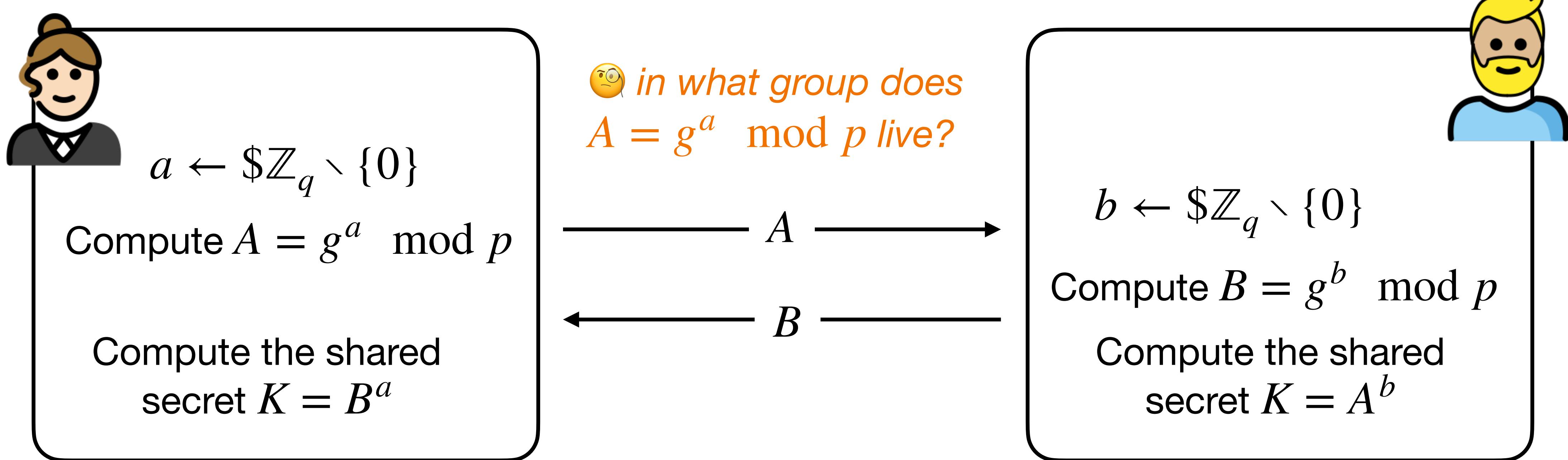
Public Key Cryptography

- Introduction
- One-Way Trapdoor Functions
- The Hash-and-Sign Paradigm

The Diffie-Hellman Key Exchange Protocol (DH) [Textbook]

Setting

Let p be a large prime (2048-bits long). Find a generator g of a subgroup of prime order q in \mathbb{Z}_p^* . Let p, q, g be all public information.



Security (intuitions)

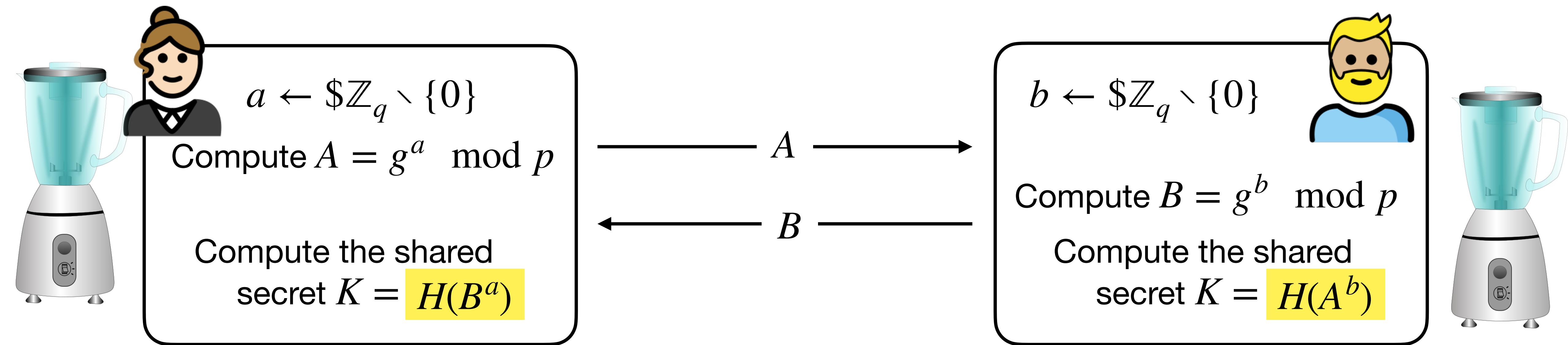
The values a, b should not be obtainable from A, B

The value K should not be computable from A, B

Given A, B , the value K should be indistinguishable from random

...we will discuss security assumptions in detail later in this lecture

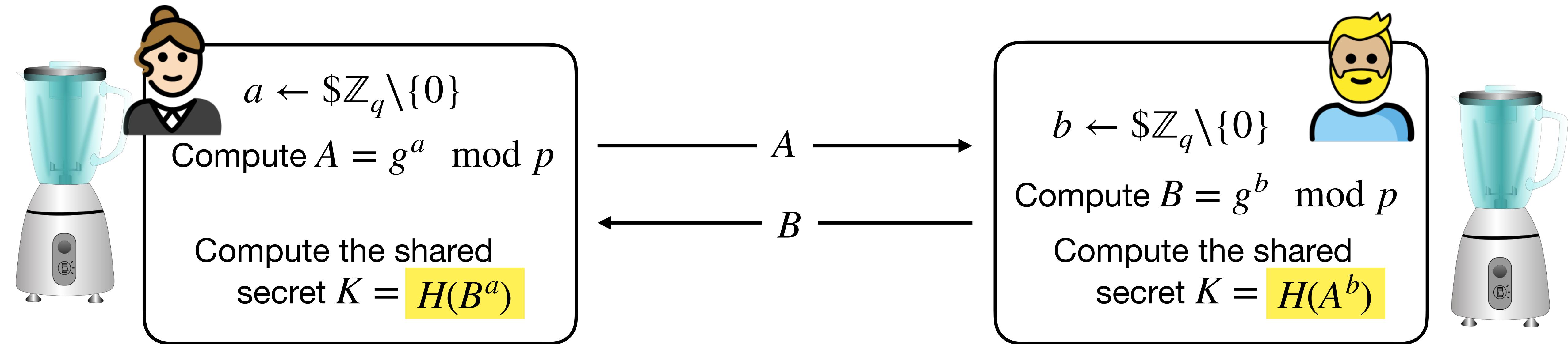
The Diffie-Hellman Key Exchange Protocol



For a number of reasons, it is best to *hash/process* the textbook DH key prior to use!

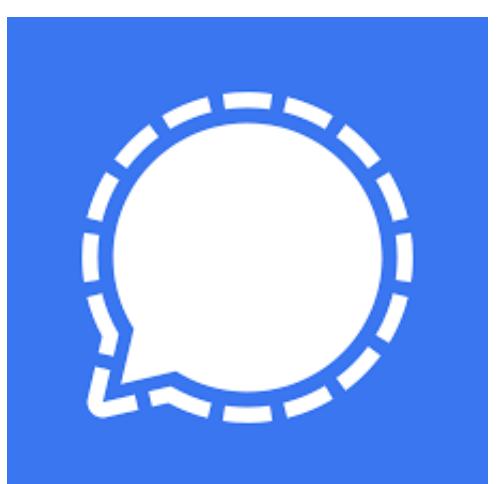
1. It is (computationally) easy to find the least significant bit (LSB) of K
2. $dLog_g(K)$ is even iff K is *quadratic residue* in \mathbb{Z}_p^* .
3. It is easy to compute the s LSBs or MSBs of K when $p - 1 = 2^s \cdot q$, with q odd.
("Hardness of Distinguishing the MSB or LSB of Secret Keys in Diffie-Hellman Schemes" [FPSZ06])
4. Computing the $s + 1$ -th LSB or MSB of K , when $p - 1 = 2^s \cdot q$, with q odd is hard
5. Heuristically $H(A^b)$ is a good key if $H : \{0,1\}^{|p|} \rightarrow \{0,1\}^{256}$ is a cryptographic hash function

The Diffie-Hellman Key Exchange Protocol



For a number of reasons, it is best to *hash/process* the textbook DH key prior to use!

Moreover, there are several ways to boost security for DH and dLog:



Triple Diffie-Hellman ([X3DH](#))

\mathbb{G} is made of points on an elliptic curve



Man*-in-the-Middle Attack Against the DH Key Exchange

Alice and Bob want to find a way to share a secret key k without relying on a previously shared secret **AND** they want to do so, using a public channel, that is monitored* by the Adversary

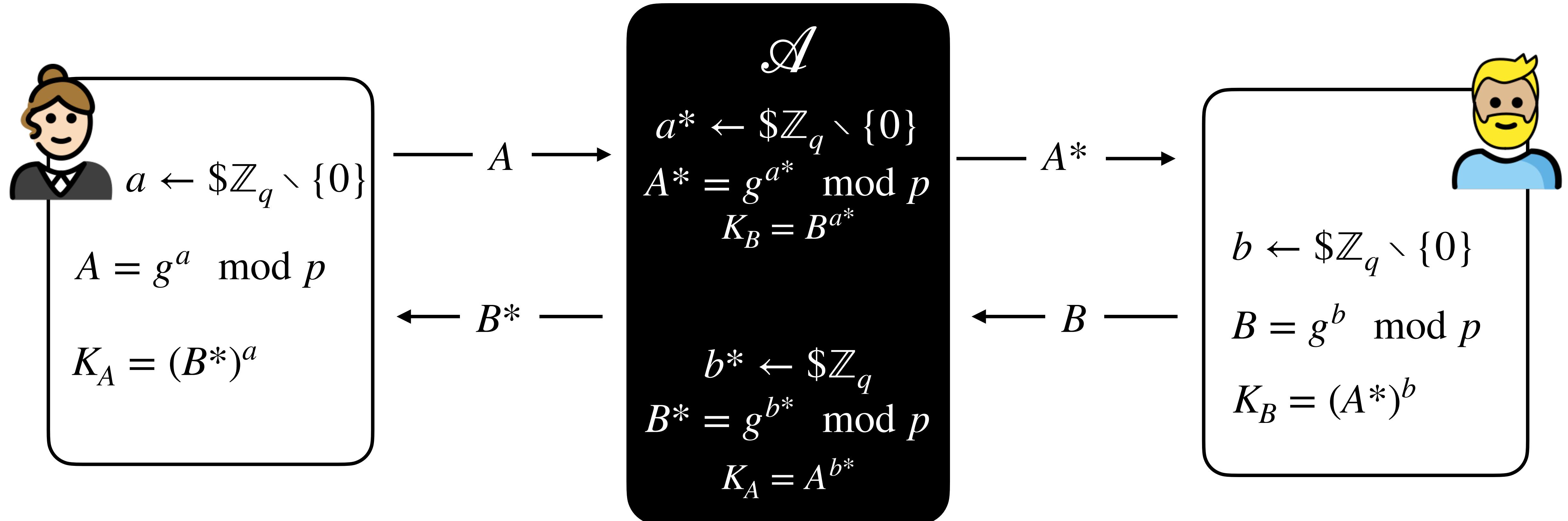


*For now we only consider passive \mathcal{A} (eavesdropper)

⚠️ What goes bad if \mathcal{A} is active?

- * I keep this wording for easy reference due to its historical adoption. Modern (and more appropriate) reformulations are:
- (1) Machine/Monster/Meddler/Manipulator in the Middle (with the same shortening MiM); or
 - (2) Person/Adversary/Attacker in the Middle

The MiM Attack Against the DH Key Exchange



⚠️ *What's enabling this attack?* DH does not authenticate whom you are doing a key exchange with



**The second main challenge that public key cryptography addresses is
(cryptographic) *Authentication***

[Next Lecture]

Lecture Agenda

Key-Exchange

- Problem Statement
- A Simple Solution
- Formalisation: **Group Theory**

Diffie-Hellman Key Exchange (DH)

- The Protocol
- On the Bit Security of DH Keys
- Securing DH Keys
- MiM Attack

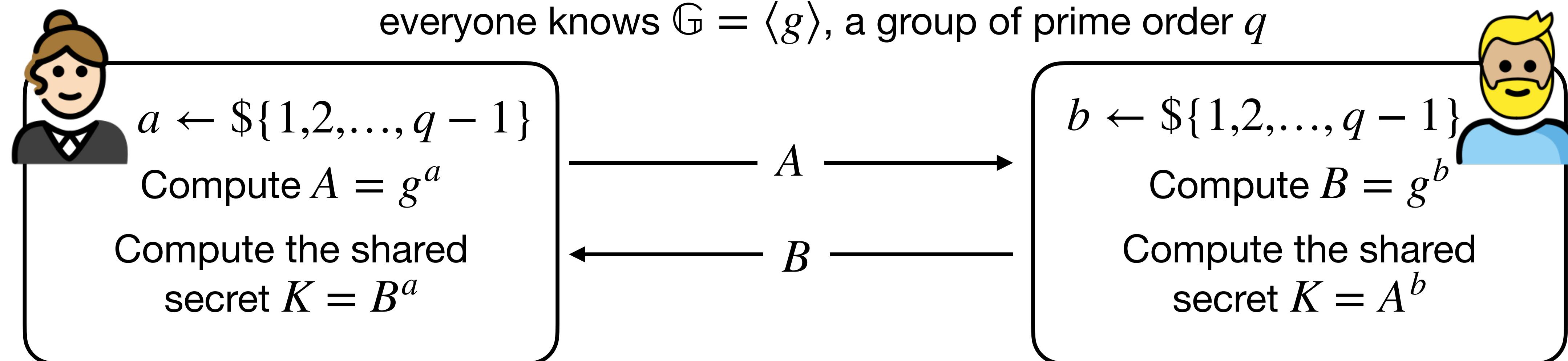
Security for DH

- DL, CDH, DDH Problems
- Relation Between Problems
- $CDH \leq DL$ (**Poof**)

Public Key Cryptography

- Introduction
- One-Way Trapdoor Functions
- The Hash-and-Sign Paradigm

Security for the DH Key Exchange



Attempt 1: it is necessary that the values a, b are not obtainable from A, B .

Discrete Logarithm Problem (DL):

Let \mathbb{G} be a group of prime order q for large prime q ($\log_2(q) = n$) and g be a generator of \mathbb{G} .

Given $A \in \mathbb{G}$ **find** a **such that** $A = g^a$.

The Discrete Logarithm Assumption

Informally: the discrete logarithm *assumption* states that DL *problem is hard*.

Let \mathbb{G} be cyclic group of order q (where q is a n -bit long prime) and g be a generator of \mathbb{G} . The **discrete logarithm assumption** states that it is computationally infeasible for any efficient attacker to find the exponent x such that $g^x = h$ for a random $h \in \mathbb{G}$.

Formally: $\Pr[x^* = x \mid x \leftarrow \mathbb{Z}_q, x^* \leftarrow \mathcal{A}(q, g, g^x)] < \text{negl}(n)$

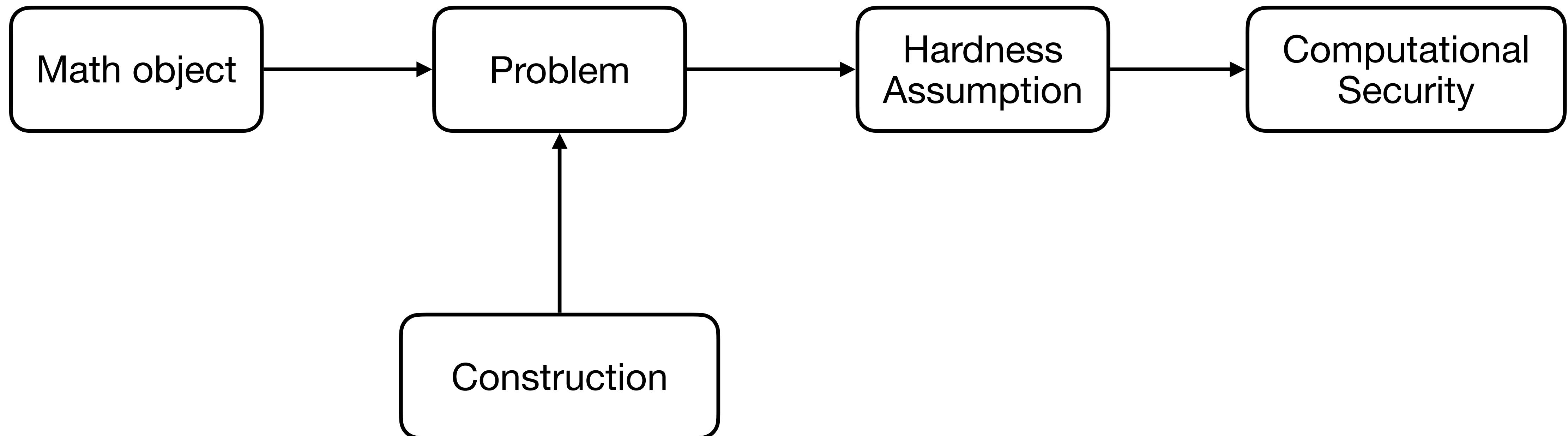
This is an **assumption**: it **cannot be proven!**

Decades (at least since 1976) of cryptanalysis and scrutiny by the cryptographic community world-wide has made us gain confidence that this assumption is true, for *large enough* primes

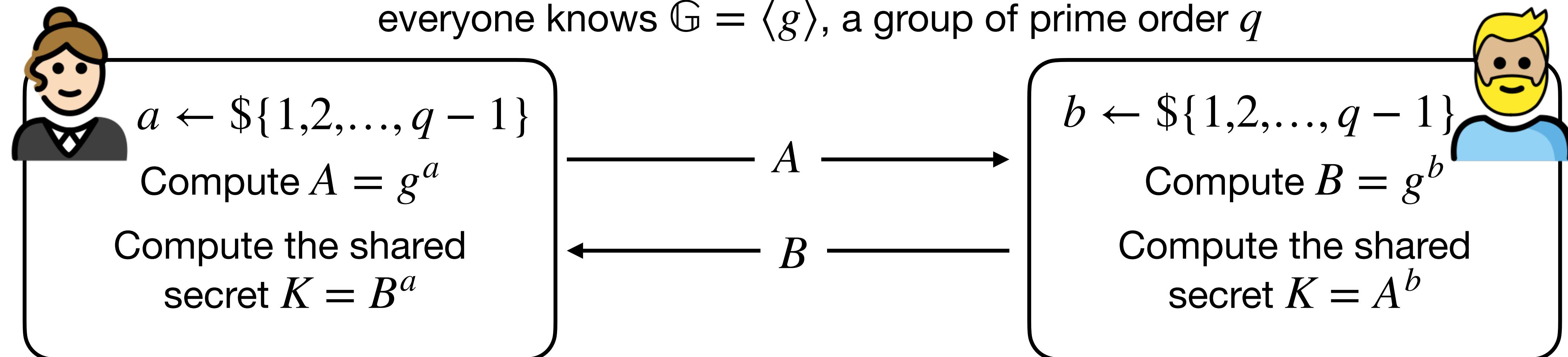
A silhouette of a person climbing a steep rock face against a backdrop of a colorful, cloudy sky at sunset or sunrise. The climber is shown from the side, pulling themselves up with ropes and carabiners. The rock face is dark and textured.

Why Do We Need Assumptions in Cryptography?

The Flow Chart of How Cryptographic Scheme Are Born



Security for the DH Key Exchange



Attempt 1: it is necessary that the values a, b are not obtainable from A, B .

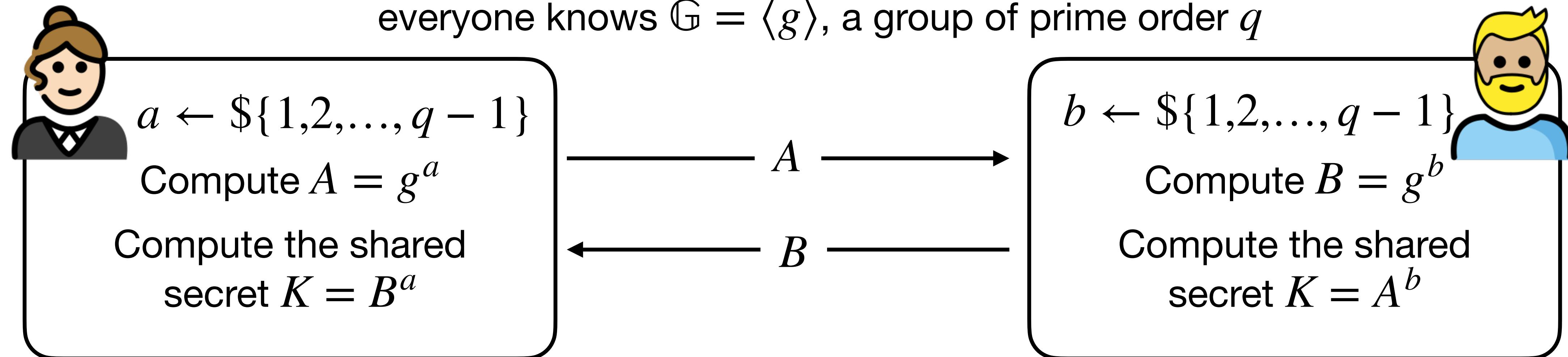
Discrete Logarithm Problem (DLP):

Let \mathbb{G} be a group of prime order q for large prime q ($\log_2(q) = n$) and g be a generator of \mathbb{G} .

Given $A \in \mathbb{G}$ **find** a **such that** $A = g^a$.

..it may still be possible for \mathcal{A} to compute K combining A, B and without learning a, b ..

Security for the DH Key Exchange



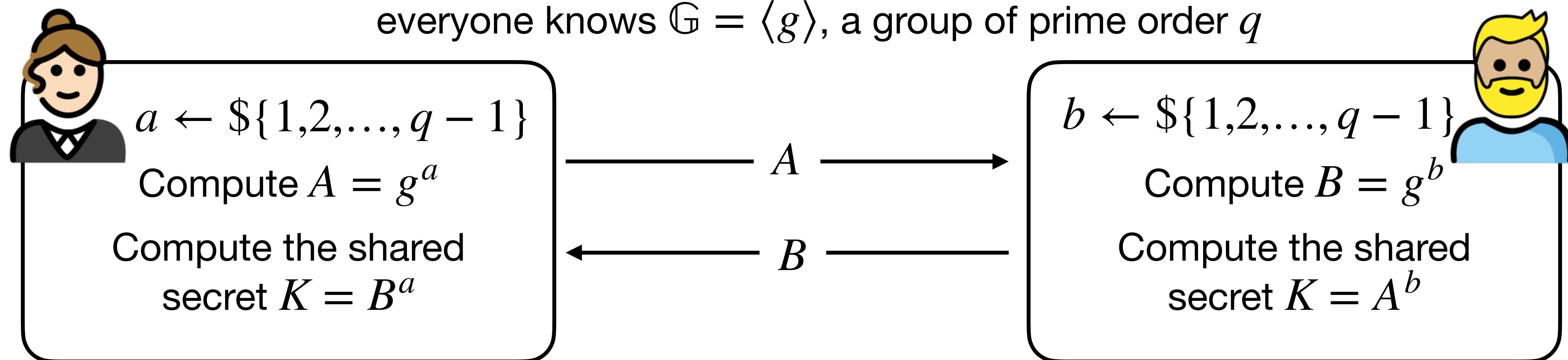
Attempt 2 (final): it should be infeasible to compute K combining A, B .

Computational Diffie-Hellman Problem (CDH):

Let \mathbb{G} be a group of prime order q for large prime q ($\log_2(q) = n$) and g be a generator of \mathbb{G} .

Given $A = g^a, B = g^b \in \mathbb{G}$ **find** $K \in \mathbb{G}$ **such that** $K = g^{ab}$.

Security for the DH Key Exchange



Attempt 3 (refinement): it should be infeasible to compute distinguish whether K or $K' \neq K$ was used as a shared key.

Decisional Diffie-Hellman Problem (DDH):

Let \mathbb{G} be a group of prime order q for large prime q ($\log_2(q) = n$) and g be a generator of \mathbb{G} .

Given $A = g^a$, $B = g^b$, $C \in \mathbb{G}$ **determine whether** $C = g^{ab}$ **or not.**

Three Problems

Discrete Logarithm Problem (DL):

Given $A \in \mathbb{G}$ find a such that $A = g^a$.

VI Proof

Computational DH Problem (CDH):

Given $A = g^a$ and $B = g^b$ find
 $K = g^{ab}$

Decisional DH Problem (DDH):

Given $A = g^a, B = g^b, C \in \mathbb{G}$
determine if $C = g^{ab}$ or not.

... And Their Relations

Let A and B be two computational problems. A is said to **efficiently reduce** to B , written $A \leq B$ if:

- There is an algorithm which solves A using an algorithm which solves B .
- This algorithm runs in polynomial time if the algorithm for B does.

Proof structure: build a **reduction** (sequence of steps, program)

- Assume we have an oracle (or efficient algorithm) to solve problem B.
- We then use this oracle to give an efficient algorithm for problem A.

For more info, check out [this blog](#)

Lecture Agenda

Key-Exchange

- Problem Statement
- A Simple Solution
- Formalisation: **Group Theory**

Diffie-Hellman Key Exchange (DH)

- The Protocol
- On the Bit Security of DH Keys
- Securing DH Keys
- MiM Attack

Security for DH

- DL, CDH, DDH Problems
- Relation Between Problems
- $\text{CDH} \leq \text{DL}$ (**Poof**)

Public Key Cryptography

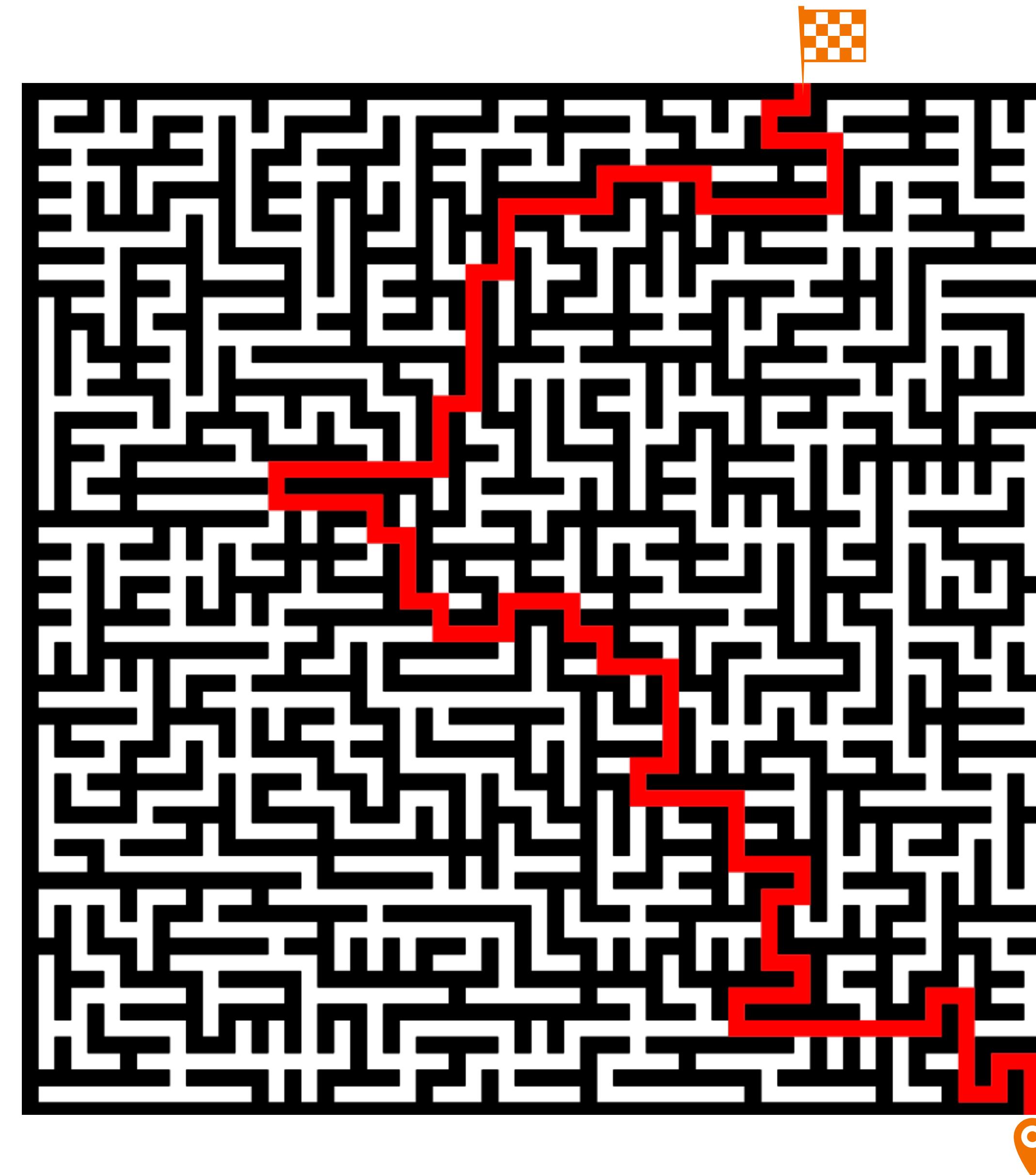
- Introduction
- One-Way Trapdoor Functions
- The Hash-and-Sign Paradigm

The One Fundamental Concept in Public Key Cryptography (PKC)

This is a **hard** problem
..unless..

🧐 *What does “hard” mean in cryptography?*

[the problem is solvable, but solving it requires time proportional to the age of our universe]



... you know some additional information that makes solving the problem easy!
(trapdoor)

PKC is all about this ‘efficiency gap’ in solving a mathematical problem

+ tons of randomness

One-Way Functions [Recap]



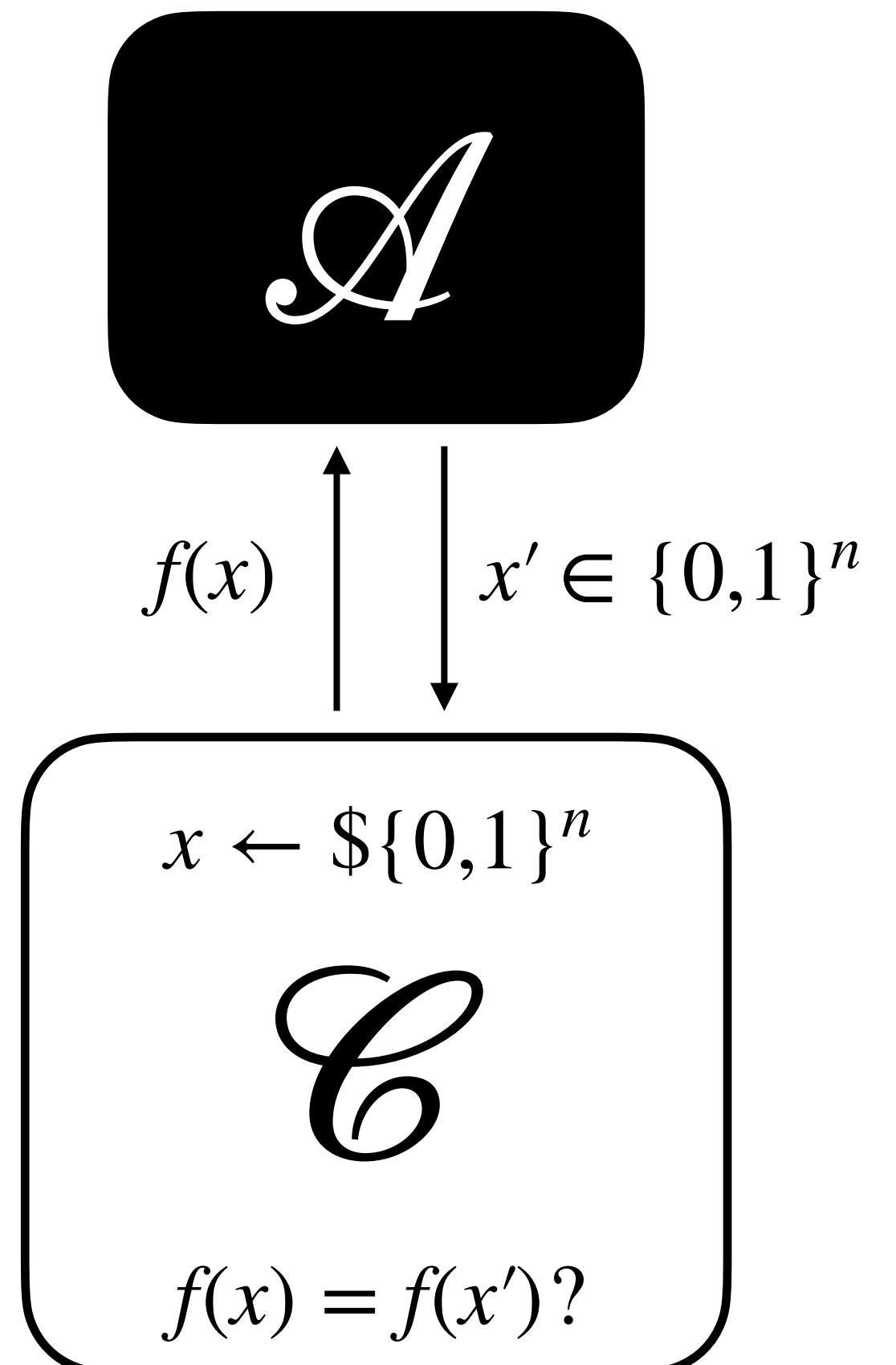
Definition: ONE-WAY FUNCTION

A function $f: \{0,1\}^n \rightarrow \{0,1\}^d$ is one-way if:

- (1) There exists an algorithm that computes $f(x)$ in **polynomial time** for all inputs $x \in \{0,1\}^n$ (f is efficiently computable)
- (2) For every PPT algorithm \mathcal{A} there is a **negligible** function $negl_{\mathcal{A}}(\cdot)$ such that for sufficiently large values of $n \in \mathbb{N}$ it holds that

$$Pr[f(x) = f(x') \mid x \leftarrow \$\{0,1\}^n, x' \leftarrow \mathcal{A}(f(x))] \leq negl_{\mathcal{A}}(n)$$

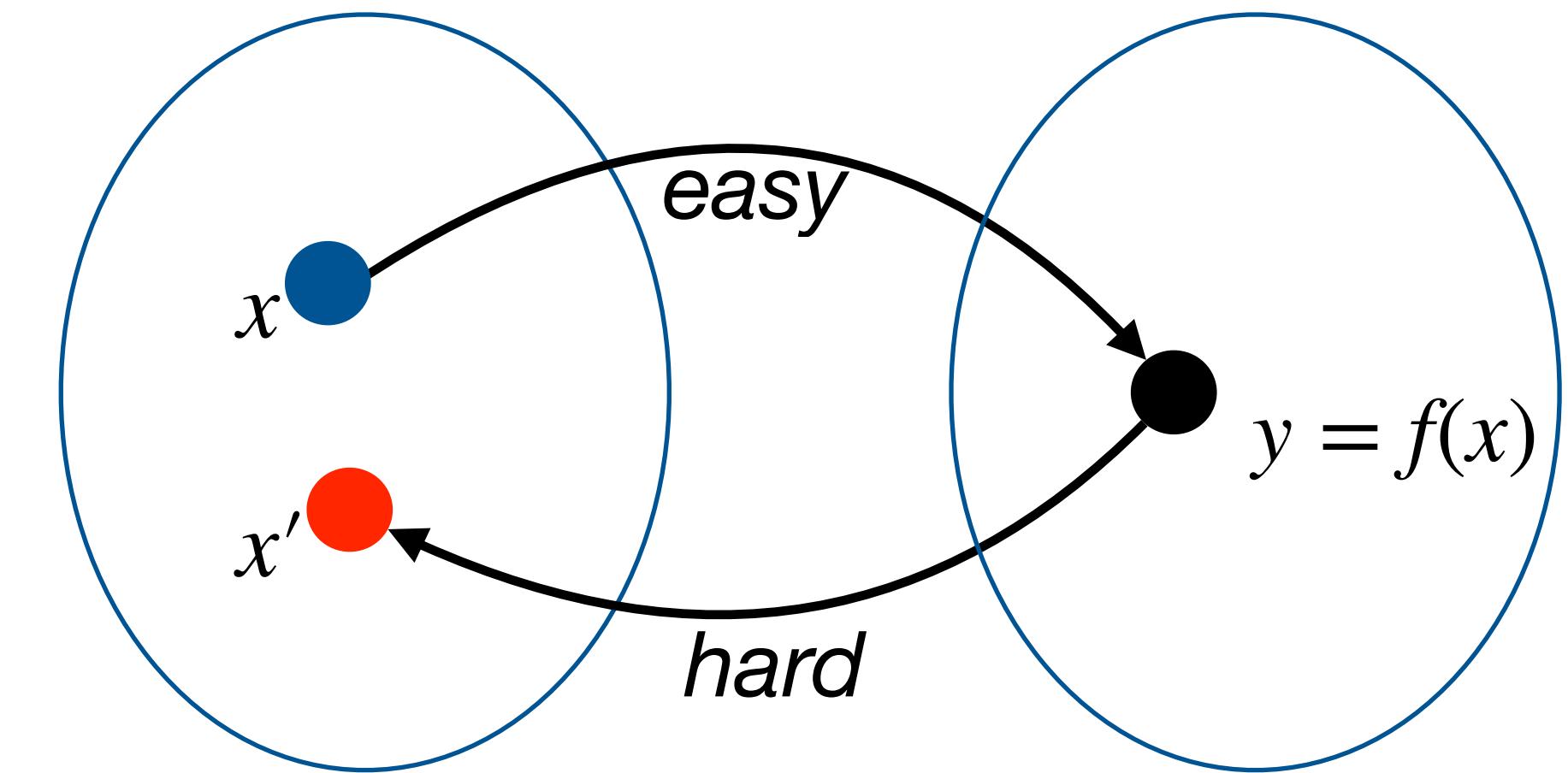
win / lose ←



OWFs Examples (Algebraic Problems Presumed To Be “Hard”)

Integer Factorisation $f(p, q) = N = p \cdot q$

given a large composite number N it is computationally hard to find p, q (if p, q are large *safe* primes).



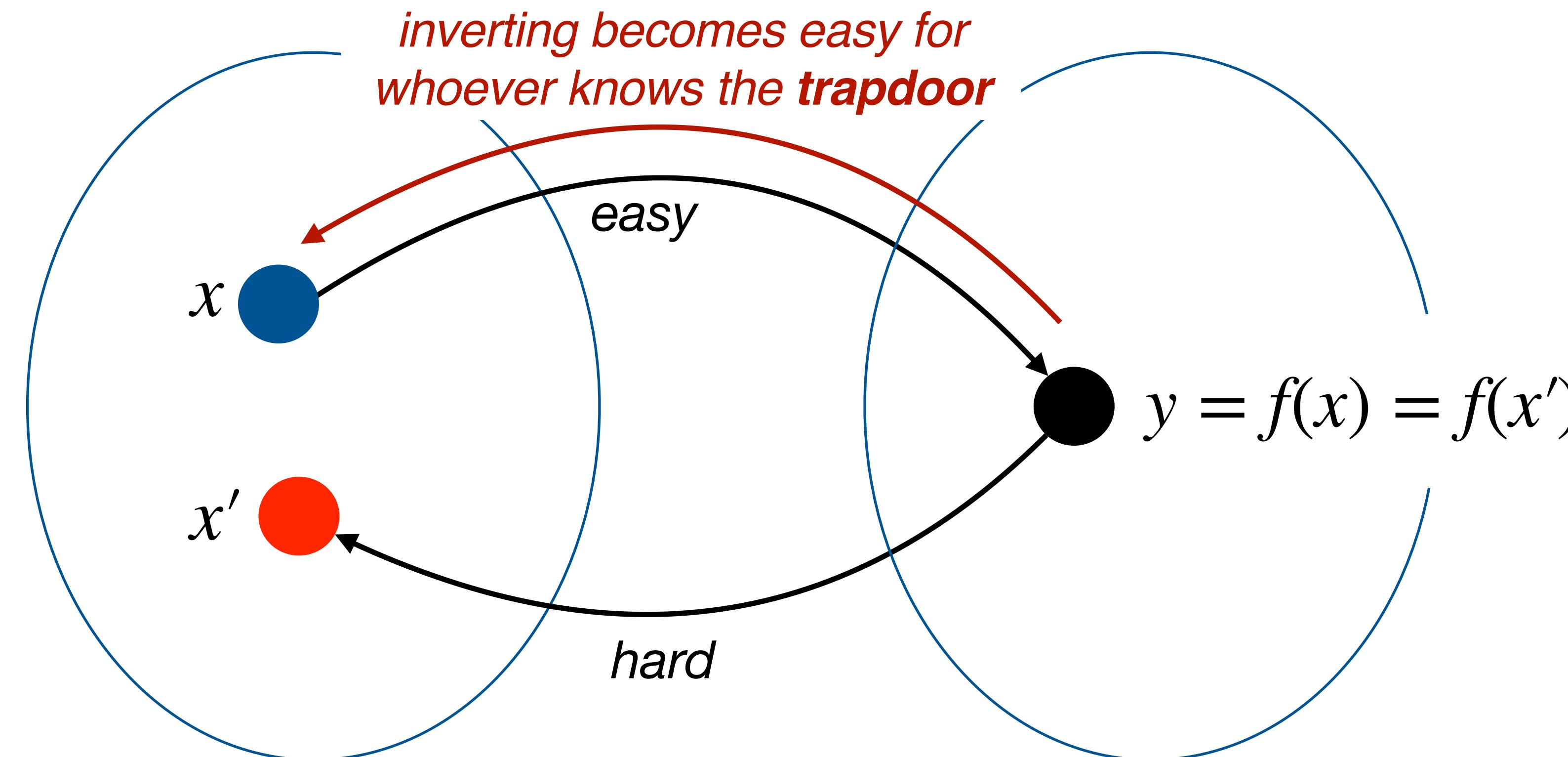
RSA (Assumption): $f_{N,e}(x) = y$ ($= x^e \pmod{N}$)

Given (N, e, y) such that $N = pq$, $\log_2(p) = \log_2(q) = n$, $e \in \mathbb{Z}_{\varphi(N)}^*$, and $y \in \mathbb{Z}_N^*$ any PPT adversary has only negligible probability to **find** $x \in \mathbb{Z}_N^*$ **such that** $x^e = y \pmod{N}$.

Discrete Logarithm $f(x) = h = g^x \pmod{p}$

Given $g, h \in \mathbb{Z}_p$ for a large prime p ($\log_2(p) = 2048$) it is computationally hard to find $x = \text{DL}_g(h)$.

Trapdoor One-Way Function (Aka One-Way Trapdoor Functions)



One-Way Trapdoor Functions - Examples

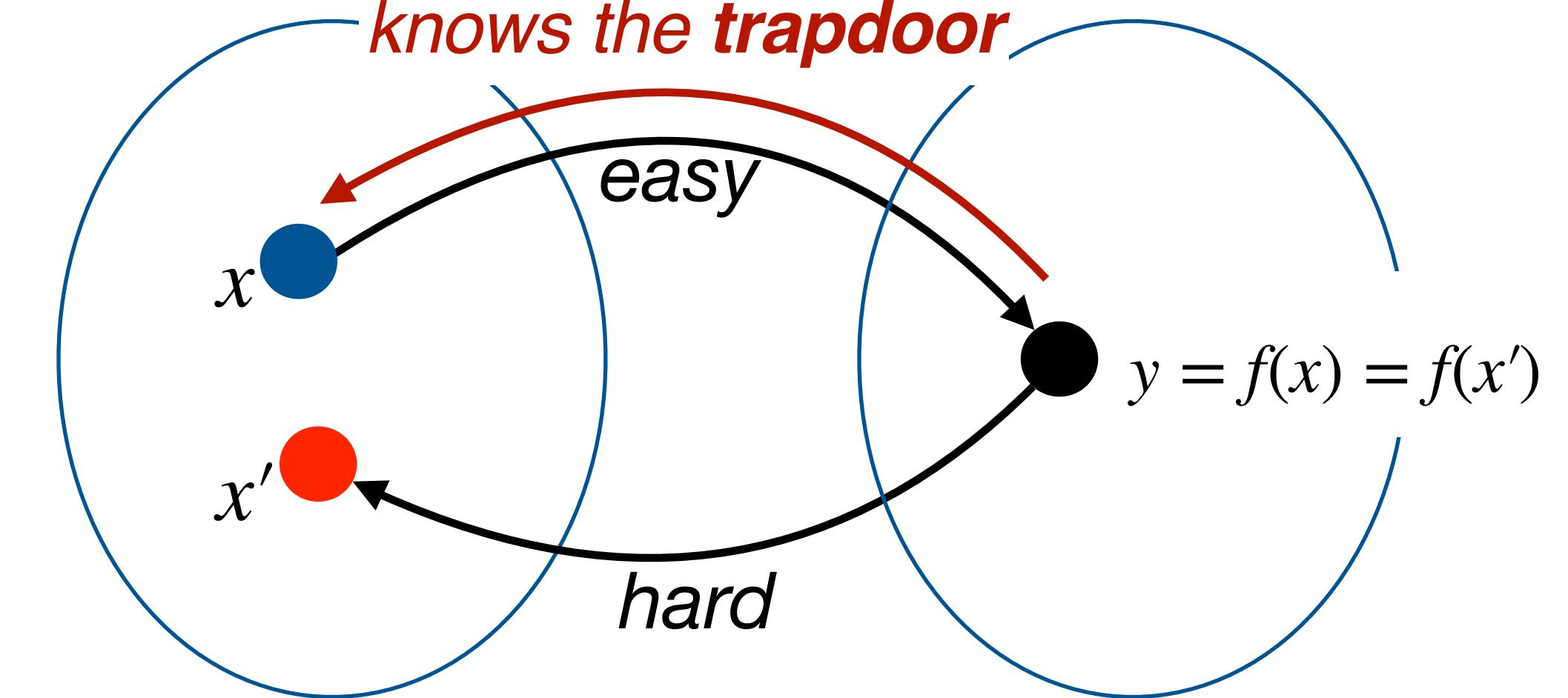
Integer Factorisation $f(p, q) = N = p \cdot q$

Trapdoor = (p, q)

Discrete Logarithm $f(x) = h = g^x \pmod{p}$

Trapdoor = x

*inverting becomes
easy for whoever
knows the trapdoor*



RSA: $f_{N,e}(x) = y$ ($= x^e \pmod{N}$)

Trapdoor = d the inverse of e modulo $\varphi(N) = (p-1)(q-1)$

Recap From the Last Lecture

Kerckhoffs's Principle

Group Theory

$\mathbb{Z}_p, \mathbb{Z}_p^*, \varphi(N)$

Euler & Fermat Theorems

Corollary $a^x \pmod{N} = a^{x \pmod{\varphi(N)}} \pmod{N}$

DH Key Exchange

DL, CDH, DDH

MiM Attack against DH

PROOF
CDH reduces to DL

Trapdoor One-Way Functions (TDF)

Lecture Agenda

Public Key Cryptography

- Algebraic Trapdoors, Examples
- Trapdoor Permutations
- RSA Is Permutation

Digital Signatures

- Definition
- EUF-CMA
- The RSA Signature
- ECDSA (Good and Bad)
- The Hash-and-Sign Paradigm

Algebraic Trapdoors - Examples

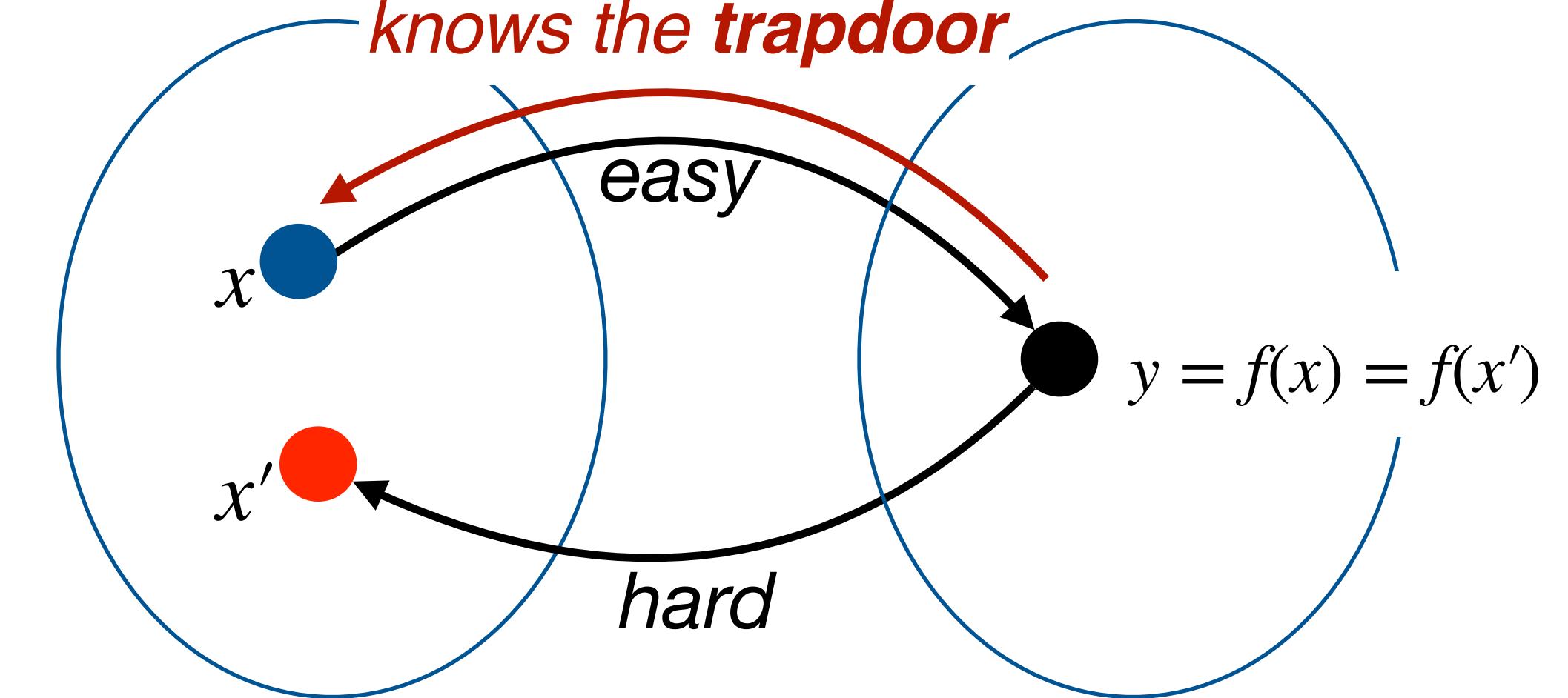
Integer Factorisation $f(p, q) = N = p \cdot q$

Trapdoor = (p, q)

Discrete Logarithm $f(x) = h = g^x \pmod{p}$

Trapdoor = x

*inverting becomes
easy for whoever
knows the trapdoor*



RSA: $f_{N,e}(x) = y$ ($= x^e \pmod{N}$)

Trapdoor = d the inverse of e modulo $\varphi(N) = (p-1)(q-1)$

(One-Way) Trapdoor Permutations

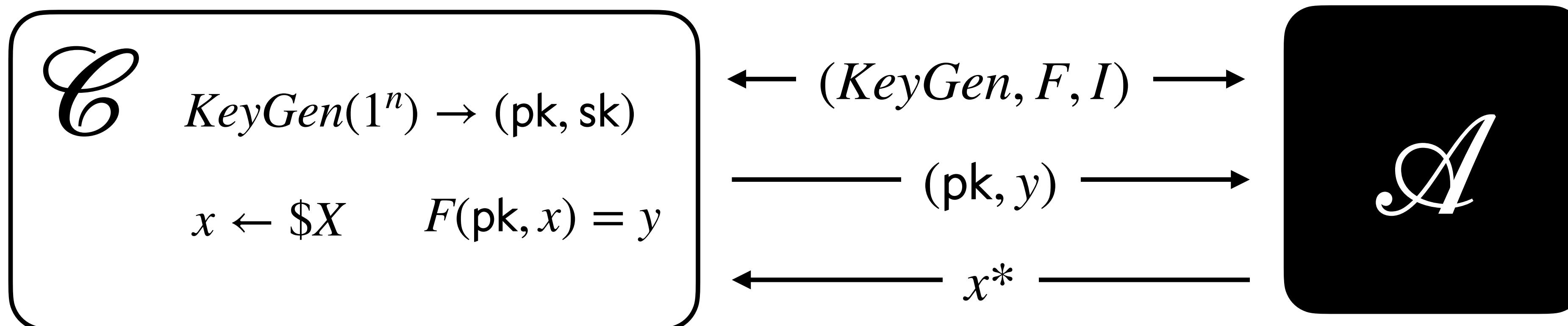
Definition: Trapdoor Permutation

A (one-way) **trapdoor permutation** defined over two finite sets X, Y is a triple of PPT algorithms $(KeyGen, F, I)$ defined as follows:

- $KeyGen(1^n) \rightarrow (\text{pk}, \text{sk})$ is a randomized key generation algorithm
- For every pk output by $KeyGen$, $F(\text{pk}, \cdot) : X \rightarrow Y$ is efficiently computable ($F(\text{pk}, x) = y$)
- For every sk output by $KeyGen$, $I(\text{sk}, \cdot) : Y \rightarrow X$ is efficiently computable ($I(\text{sk}, y') = x'$)
- $F(\text{pk}, \cdot)$ is hard to invert (without the knowledge of the corresponding trapdoor sk), formally



How can we formalise “hard to invert” as a security game?



\mathcal{A} wins if $F(\text{pk}, x^*) = y$

RSA as a Trapdoor Permutation

Theorem (RSA is a trapdoor permutation)

Let $N = p \cdot q$ (the product of two safe primes) be an integer and $e \in \mathbb{Z}_{\varphi(N)}^*$.
The function $f_{N,e}(x) = x^e \pmod{N}$ is a trapdoor permutation over \mathbb{Z}_N^* .

Proof

- $\text{KeyGen}(1^n) \rightarrow (\text{pk} = (N, e), \text{sk} = (p, q, d))$ with $N = pq$, $e \in \mathbb{Z}_{\varphi(N)}^*$, $d = e^{-1} \pmod{\varphi(N)}$
- $F(\text{pk}, x) = x^e \pmod{N}$ is efficiently computable (binExp, Euler's theorem, modular arithmetic)
- $I(\text{sk}, y) = y^d \pmod{N}$ is eff.com. given the trapdoor d , and I is an inverse for F over \mathbb{Z}_N^* :
$$I(\text{sk}, F(\text{pk}, x)) = (x^e \pmod{N})^d \pmod{N} = x^{ed} \pmod{N} = x^{ed} \pmod{\varphi(N)} \pmod{N} = x^1 \pmod{N}$$
- F is computationally hard to invert given only $\text{pk} = (N, e)$ by the **RSA assumption**.

RSA Assumption:

Given $N = p \cdot q$ (the product of two safe primes), $e \in \mathbb{Z}_N^*$, and $y \in \mathbb{Z}_N$
it is computationally infeasible to find $x \in \mathbb{Z}_N$ such that $x^e = y \pmod{N}$.

Lecture Agenda

Public Key Cryptography

- Algebraic Trapdoors, Examples
- Trapdoor Permutations
- RSA Is Permutation

Digital Signatures

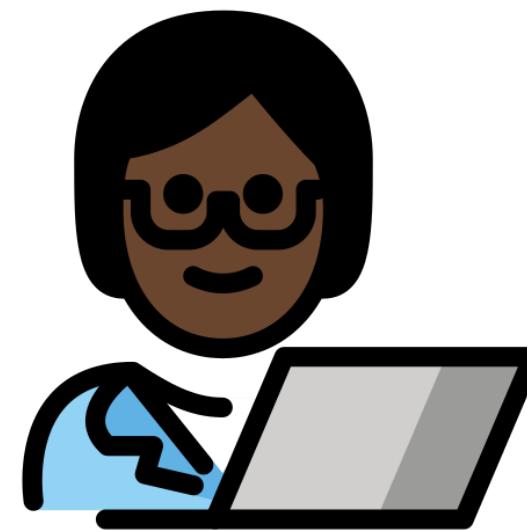
- Definition
- EUF-CMA
- The RSA Signature
- ECDSA (Good and Bad)
- The Hash-and-Sign Paradigm

Authenticating the Source of Information Over the Internet

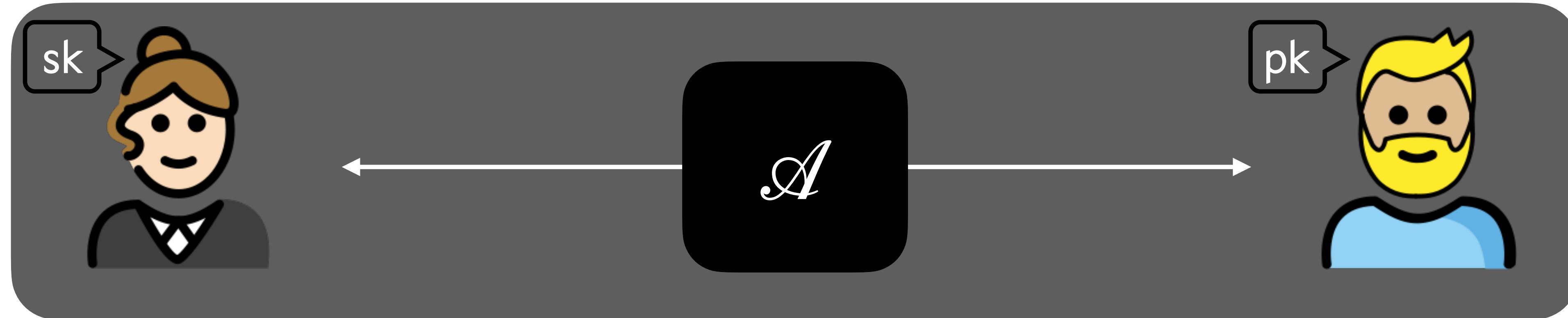
Symmetric cryptography



Problem: if both Alice and Bob know k , then cryptographically they are the same person. Bob cannot convince a third party that it was Alice producing something (e.g. a MAC) that requires the knowledge of k . *Whatever Alices produces, Bob can produce it as well!*



Asymmetric cryptography



With **public key cryptography** Alice is the only entity to know sk . If she uses it to do something that is (computationally) impossible to do without sk , then everyone can be convinced she did it.

Digital Signature - Syntax

Definition: Digital Signature

A digital signature scheme is a triple of PPT algorithms (KeyGen , Sign , Ver) defined as follows:

- $\text{KeyGen}(1^n) \rightarrow (\text{pk}, \text{sk})$ is a probabilistic key generation algorithm
- $\text{Sign}(\text{sk}, m) \rightarrow \sigma$ is a (possibly) probabilistic algorithm that outputs a signature σ for a message m
- $\text{Ver}(\text{pk}, m, \sigma) \in \{0,1\}$ returns 1 if σ is accepted as a valid signature for m against pk , or 0.

Correctness

For all key pairs $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^n)$ and for all messages m , it holds that: $\text{Ver}(\text{pk}, m, \text{Sign}(\text{sk}, m)) = 1$

Towards a Security Notion for Digital Signatures

\mathcal{A}

Adversary's Power and Knowledge

Key-Only Attack: \mathcal{A} knows only the signer's pk , and therefore only has the capability of checking the validity of signatures of messages

Known Signature Attack: \mathcal{A} knows pk and sees message/signature pairs chosen and produced by the legal signer

Chosen Message Attack: \mathcal{A} knows pk and can ask the signer to sign a number of messages of the adversary's choice.

Adversary's Goal

Existential Forgery: \mathcal{A} succeeds in creating a valid signature of a new message (never seen before)

Strong Forgery: \mathcal{A} succeeds in creating a valid signature of some message of \mathcal{A} 's choice
and the signature is different from any signature seen by \mathcal{A}

Universal Forgery: \mathcal{A} is able to generate a valid signature for *any* message (but ignores sk)

Total Break: \mathcal{A} can compute the signer's secret key sk



I LOVE

The recipe for a good security notion:

1. Choose a *realistic* adversary (PPT, Quantum...)
2. Give to \mathcal{A} the strongest starting knowledge
3. Select the weakest damage to the cryptosystem
4. DONE!



CAPPUCCINO

Towards a Security Notion for Digital Signatures

Adversary's Power and Knowledge

Key-Only Attack: \mathcal{A} knows only the signer's pk , and therefore only has the capability of checking the validity of signatures of messages (*a bit unrealistic*)

Known Signature Attack: \mathcal{A} knows pk and sees message/signature pairs chosen and produced by the legal signer (*in reality, this the minimum one should assume*)

Chosen Message Attack: \mathcal{A} knows pk and can ask the signer to sign a number of messages of the adversary's choice. (*this is our standard*)

\mathcal{A}

Adversary's Goal

Existential Forgery: \mathcal{A} succeeds in creating a valid signature of a new message (never seen before)

Strong Forgery: \mathcal{A} succeeds in creating a valid signature of some message of \mathcal{A} 's choice
and the signature is different from any signature seen by \mathcal{A}

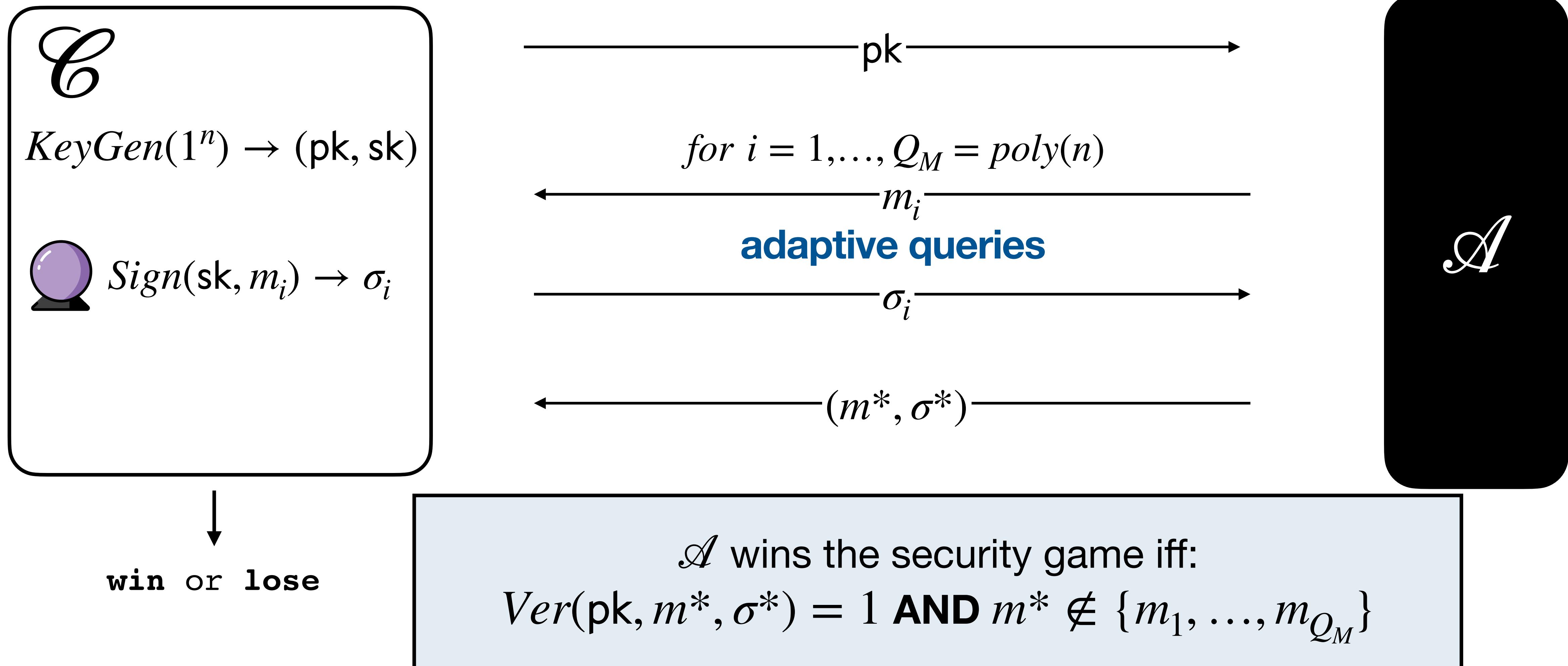
Universal Forgery: \mathcal{A} is able to generate a valid signature for *any* message (but ignores sk)

Total Break: \mathcal{A} can compute the signer's secret key sk



Existential Unforgeability Under Chosen Message Attack (EUF-CMA)

Aim: quantify the \mathcal{A} 's likelihood in forging a valid signature σ^* for a *new* message m^*



Secure Signature

A Digital Signature Scheme is said to be **secure** (unforgeable under chosen message attack) if **for all efficient** adversaries the probability that \mathcal{A} **wins** the EUF-CMA security game is **negligible**. Formally,

$$\Pr[Ver(\text{pk}, m^*, \sigma^*) = 1 \mid (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{sk}}^{\text{Sign}}}(\text{pk}) \wedge m^* \notin \{m_i\}_{i=1}^{Q_M}] \leq negl(n)$$

Example: the RSA Signature Scheme (Textbook Style)

- $\text{KeyGen}(1^n) \rightarrow (\text{pk}, \text{sk}) :$

Pick two n -bit long primes p, q and compute $N = p \cdot q$.

Pick a random invertible element $e \leftarrow \mathbb{Z}_{\varphi(N)}^*$, and compute its inverse $d \pmod{\varphi(N)}$.

Set $\text{pk} = (N, e)$ and $\text{sk} = (p, q, d)$.

- $\text{Sig}(\text{sk}, m)$: given $m \in \mathbb{Z}_N^*$, return $\sigma = m^d \pmod{N}$

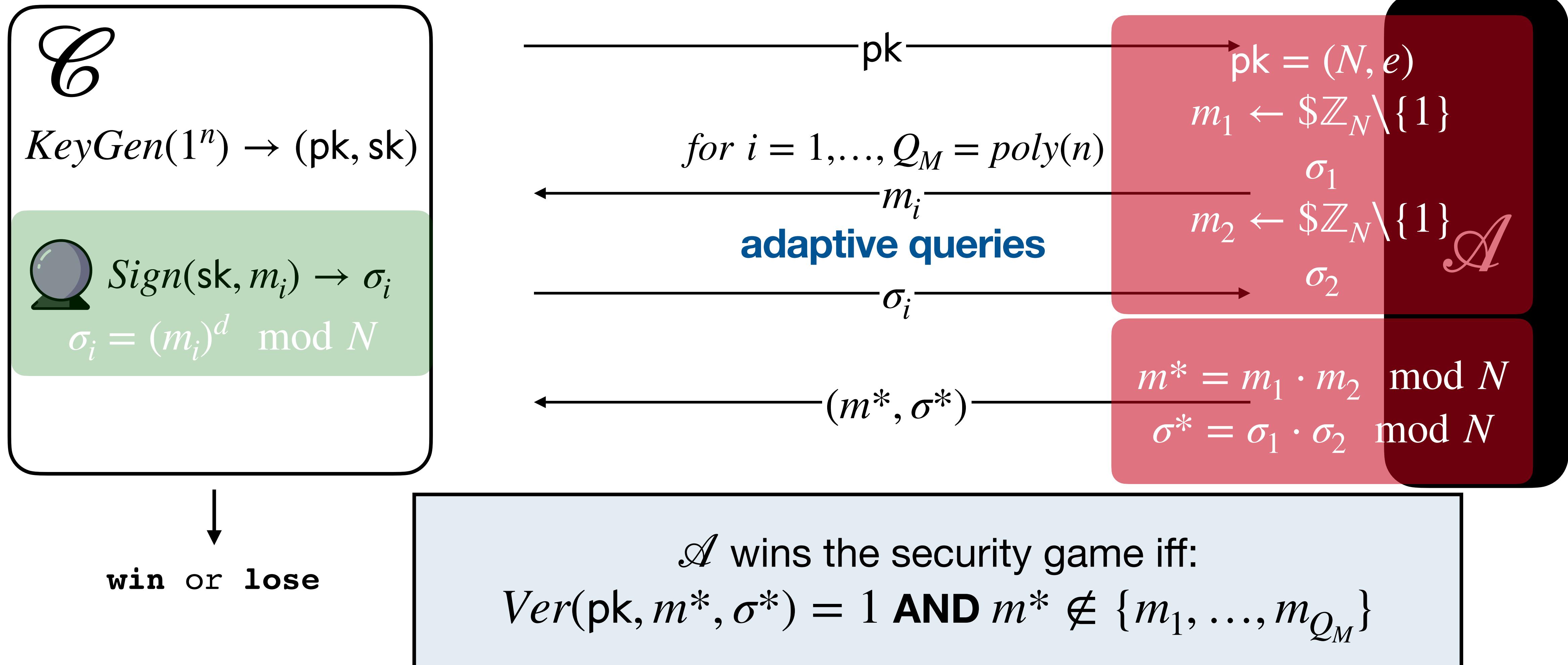
- $\text{Verify}(\text{pk}, m, \sigma)$: return 1 iff $m = \sigma^e \pmod{N}$.

For good security set $n = 1024$ (hence $\log_2(N) = 2048$)

 *How long (print form) is a number of 2048 bits?*

Is the Textbook RSA Signature EUF-CMA?

Aim: quantify the \mathcal{A} 's likelihood in forging a valid signature σ^* for a *new* message m^*



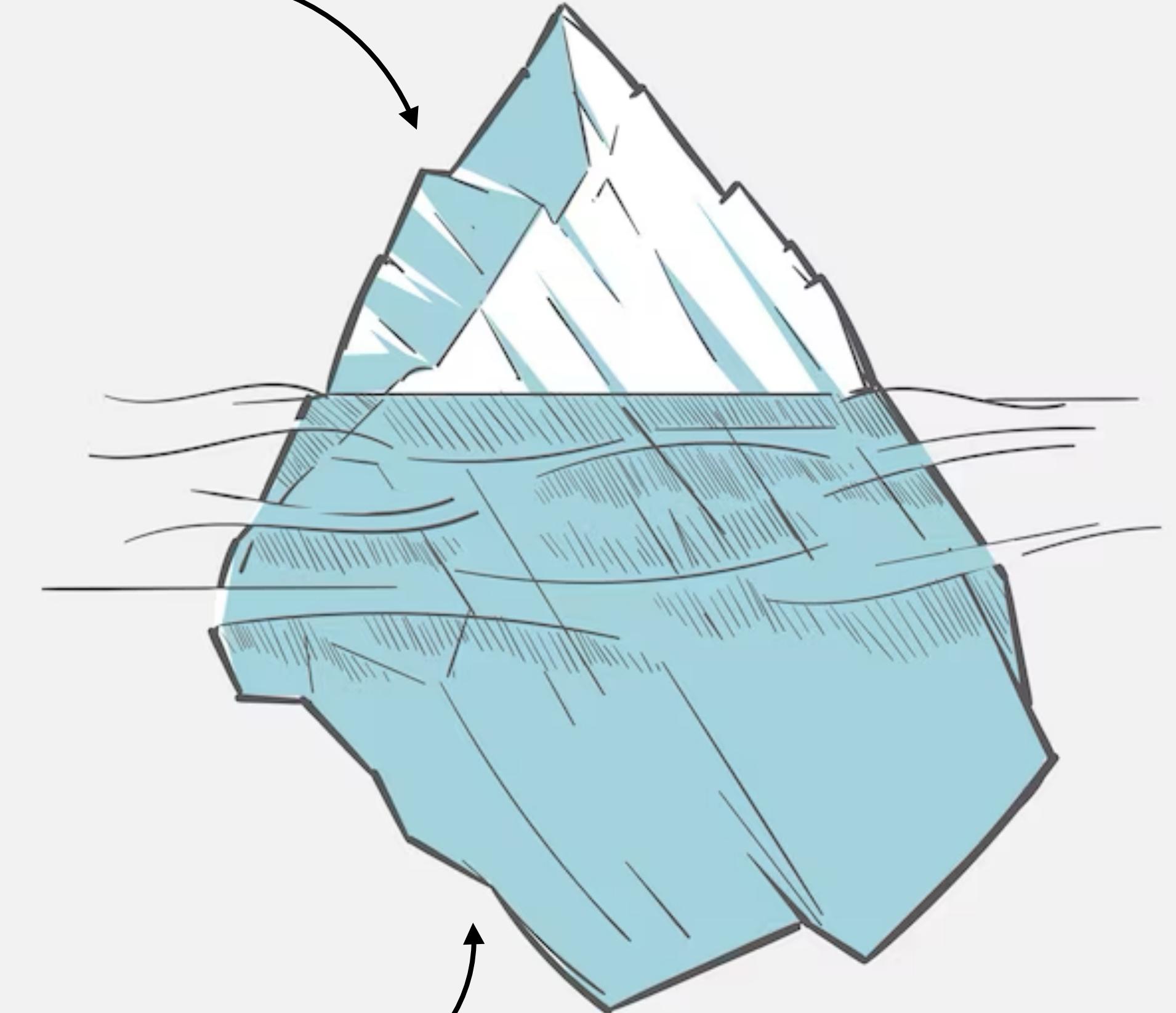
Why Do We Use RSA if It Is Not IND-CPA/EUF-CMA?

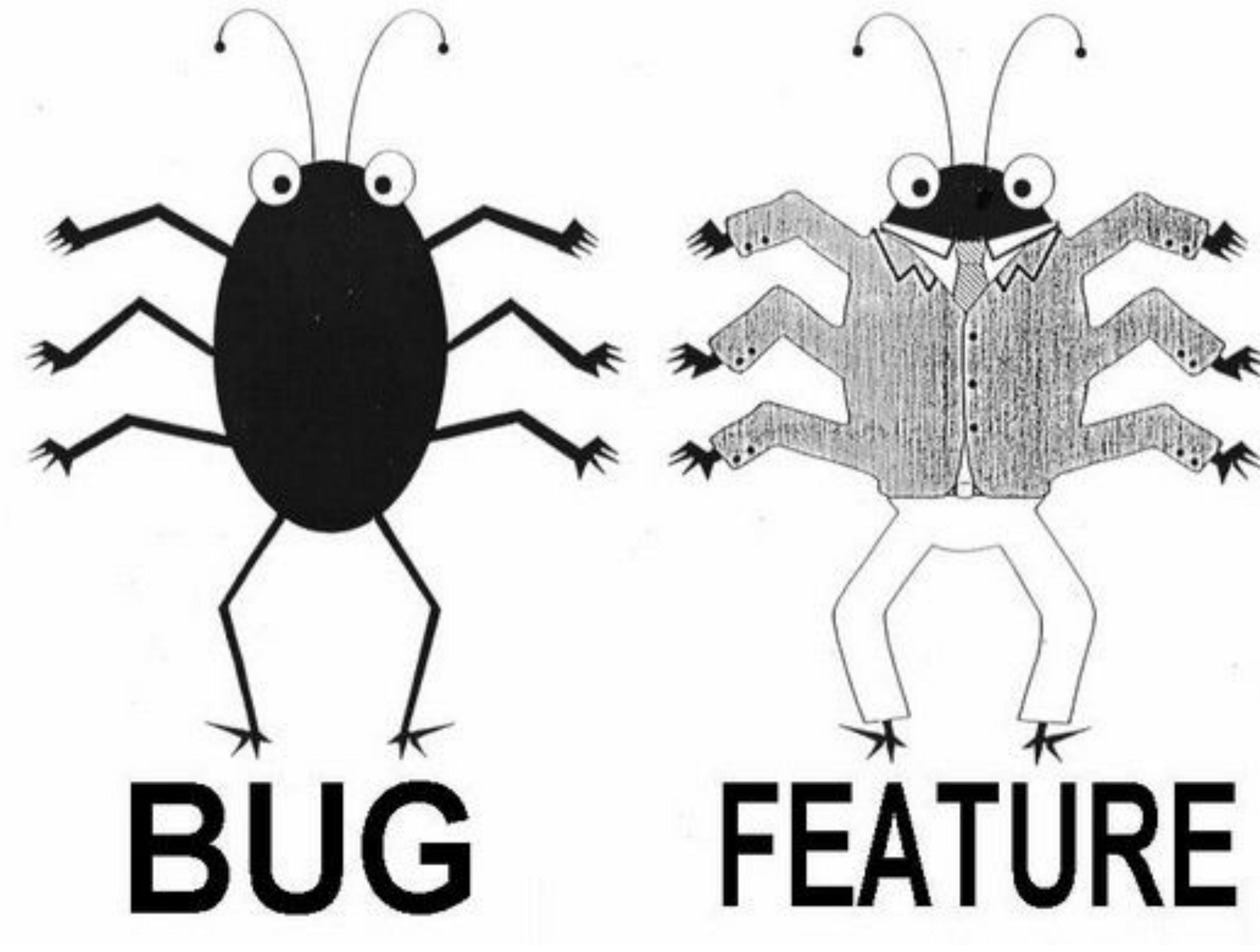
Because it is not the textbook version of the crypto system what is implemented in practice!

for EUF-CMA signing use RSA-**FDHS**,
for IND-CCA encryption use RSA-**OAEP**

Mathematics of cryptography

Real world cryptography

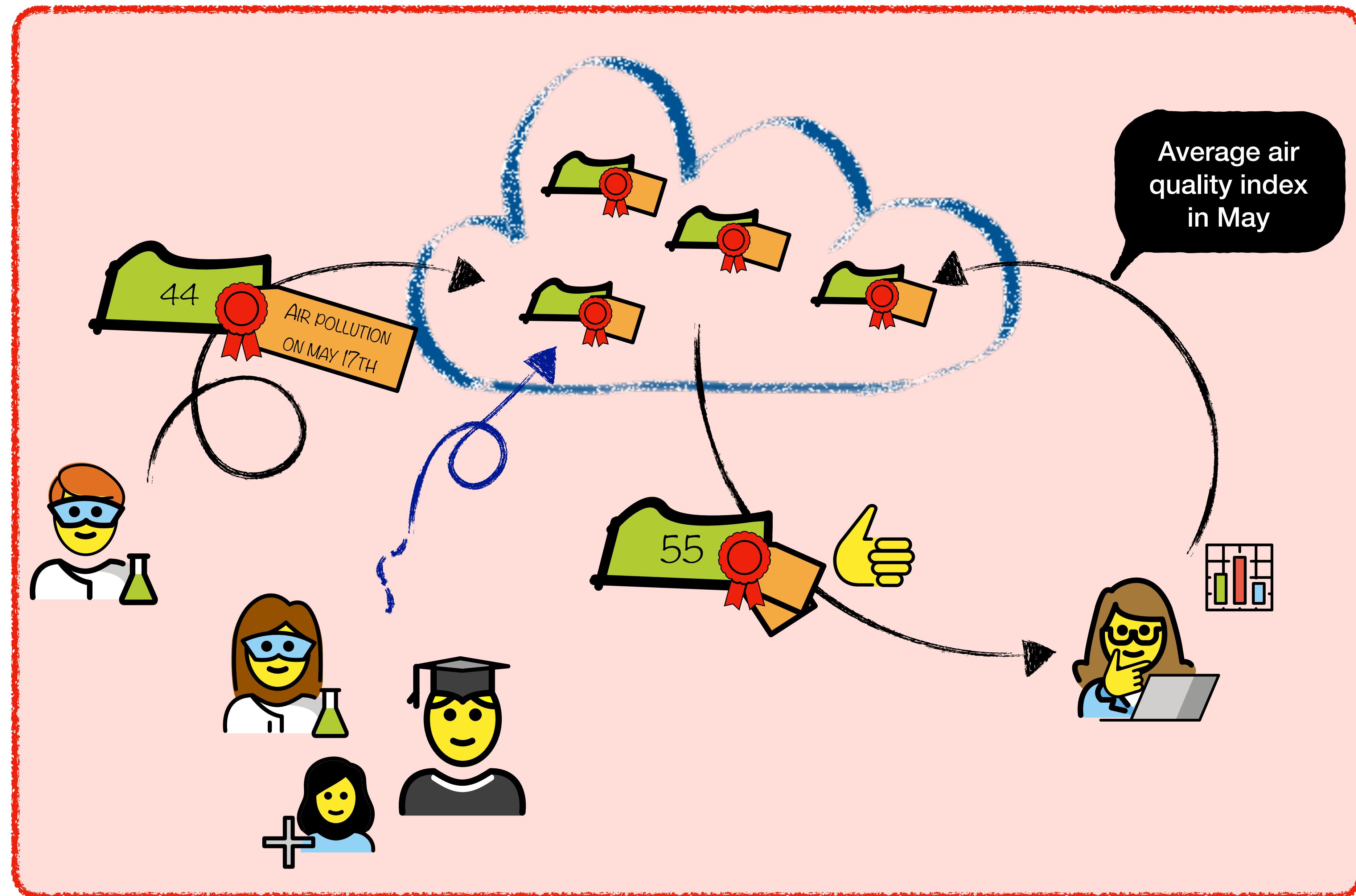




**Homomorphic
signatures can
be used to verify
the result of
outsourced
computations**

Linearly Homomorphic Signatures: What Are They Good for?

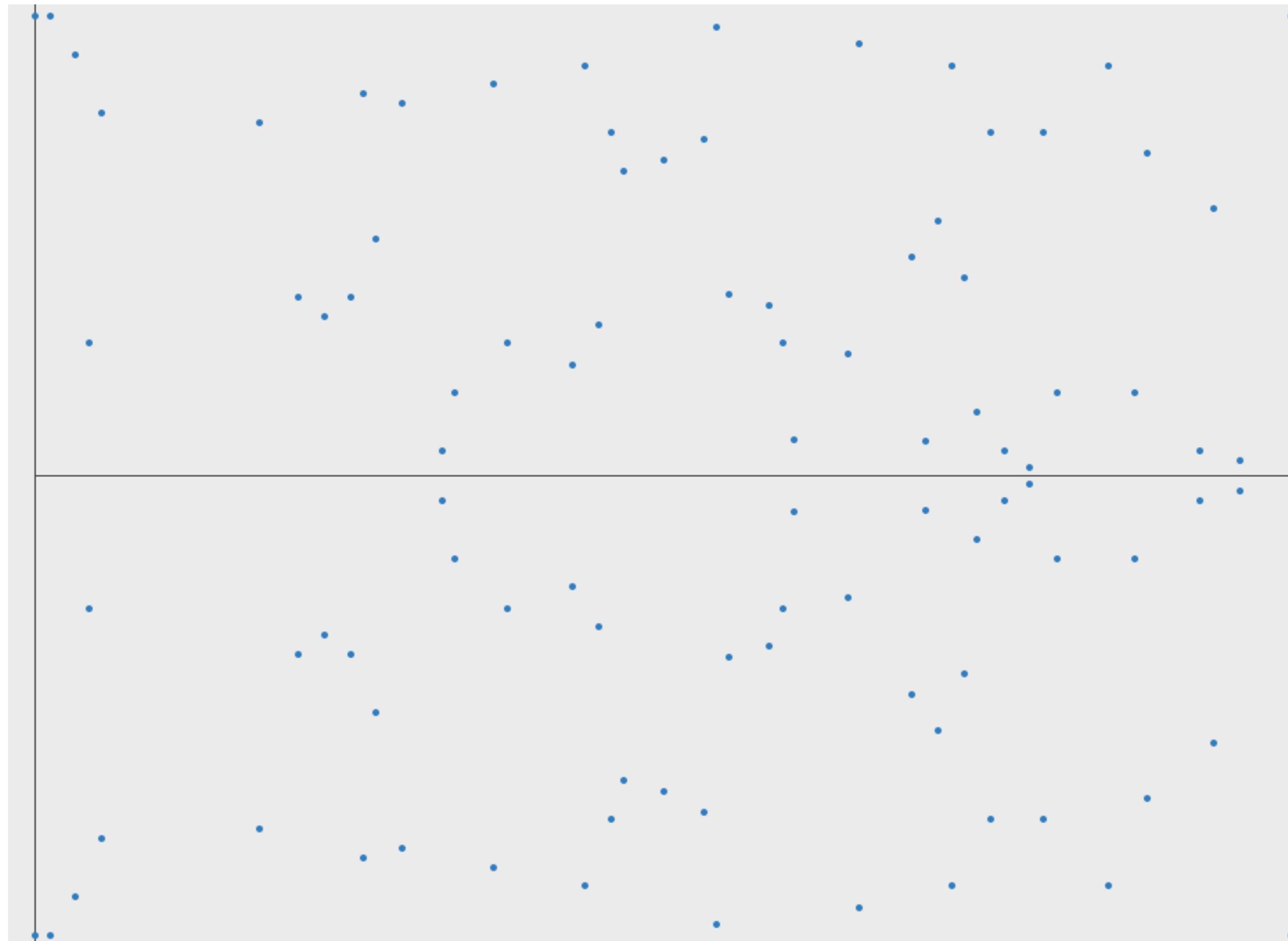
[not in exam]



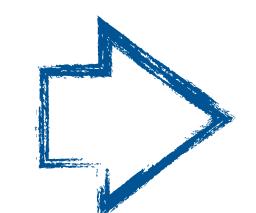
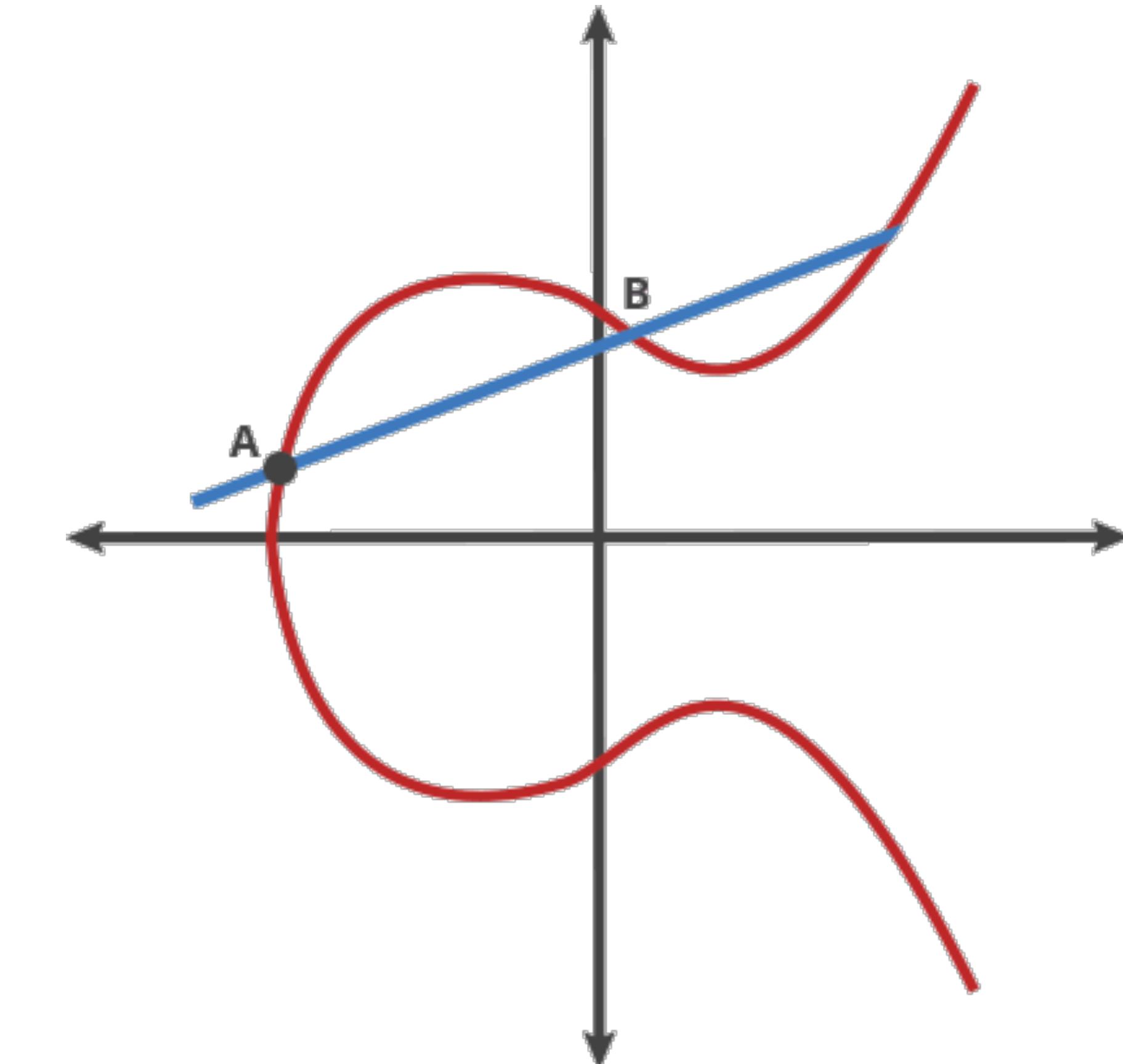
Another Very Popular Digital Signature: ECDSA

Elliptic Curve Digital
Signature Algorithm

$$y^2 = x^3 - x + 1 \pmod{97}$$



$$y^2 = x^3 + ax + b$$



*Elliptic curves have a **group** structure*

ECDSA - Algorithms

[details not in exam]

KeyGen (sec.par) \Rightarrow (sk, pk)

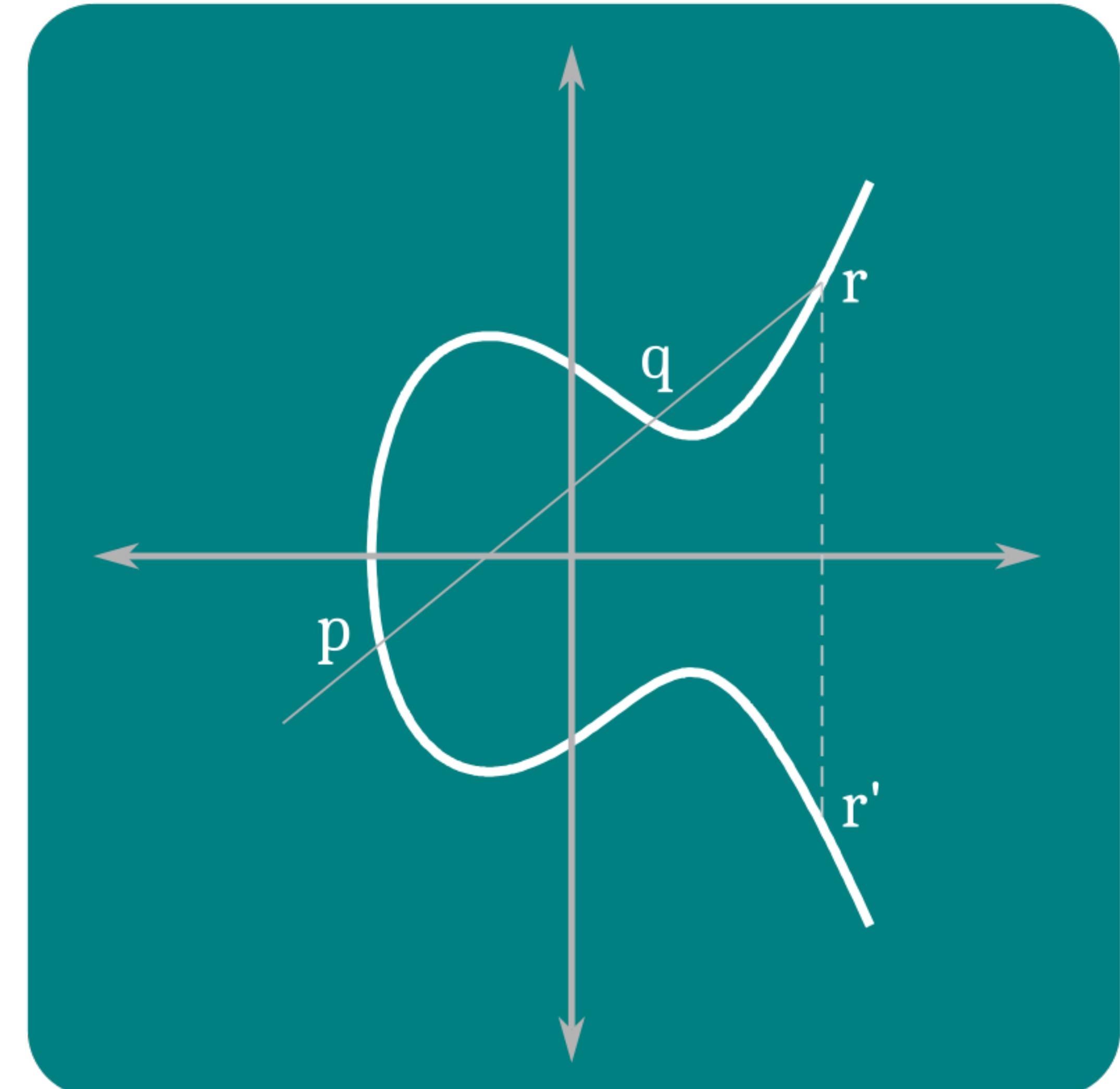
```
d ←$ [ 0 ... n-1 ]  
sk = d  
pk = Q = d*G
```

Sign (sk, msg) \Rightarrow sgn

```
k ←$ [ 0 ... n-1 ]  
R = k*G  
r = R_x mod n  
z = sha256(msg)  
s = inv(k) · (z + d · r) mod n  
sgn = (r, s)
```

Verify (pk, msg, sgn) \Rightarrow {0, 1}

```
z = sha256(msg)  
T = [z · inv(s) mod n]*G  
P = [inv(s)*r mod n]*Q  
if R == T+P return 1  
else return 0
```



ECDSA - the Good

- ★ Shorter keys and better security than the RSA signature scheme
- ★ Non malleable
- ★ IoT friendly
- ★ In wide adoption (TLS, DigiCert (Symantec), Sectigo (Comodo) ...)

ECDSA - the Bad

PS3 hacked through poor cryptography implementation

A group of hackers named fail0verflow revealed in a presentation how they ...

CASEY JOHNSTON - 12/30/2010, 6:25 PM

:(what happens if the same nonce k is used to sign two different messages?

{* SECURITY *}

Android bug batters Bitcoin wallets

Old flaw, new problem

Richard Chirgwin

Mon 12 Aug 2013 // 00:43 UTC

Sign ($sk=d$, msg) \Rightarrow sgn

$k \leftarrow \$ [0 \dots n-1]$

$R = k \cdot G$

$r = R_x \bmod n$

$z = \text{sha256}(\text{msg})$

$s = \text{inv}(k) \cdot (z + d \cdot r) \bmod n$

sgn = (r, s)

LadderLeak: Side-channel security flaws exploited to break ECDSA cryptography

Charlie Osborne 28 May 2020 at 14:07 UTC

Updated: 28 June 2021 at 09:05 UTC

Where Are Digital Signatures Used (Daily)?

```
*.github.io
[...]
Issuer Name
  Country or Region      US
  Organisation        DigiCert Inc
  Common Name          DigiCert Global G2 TLS RSA SHA256 2020 CA1
[...]
Public Key Info
  Algorithm            RSA Encryption ( 1.2.840.113549.1.1.1 )
  Parameters           None
  Public Key           256 bytes: AD 2B 14 A5 3A 4C 41 AF [...]
  Exponent             65537
  Key Size              2 048 bits
  Key Usage            Encrypt, Verify, Wrap, Derive
[...]
  Signature Algorithm   SHA-256 ECDSA
[...]
Fingerprints
  SHA-256               09 01 0C CE 9B 72 21 55 C7 E6 86 B0 77 39 D3 D2 [...]
  SHA-1                 97 D8 C5 70 0F 12 24 6C 88 BC FA 06 7E 8C A7 4D [...]
```



Part of the certificate for <https://epagnin.github.io/>

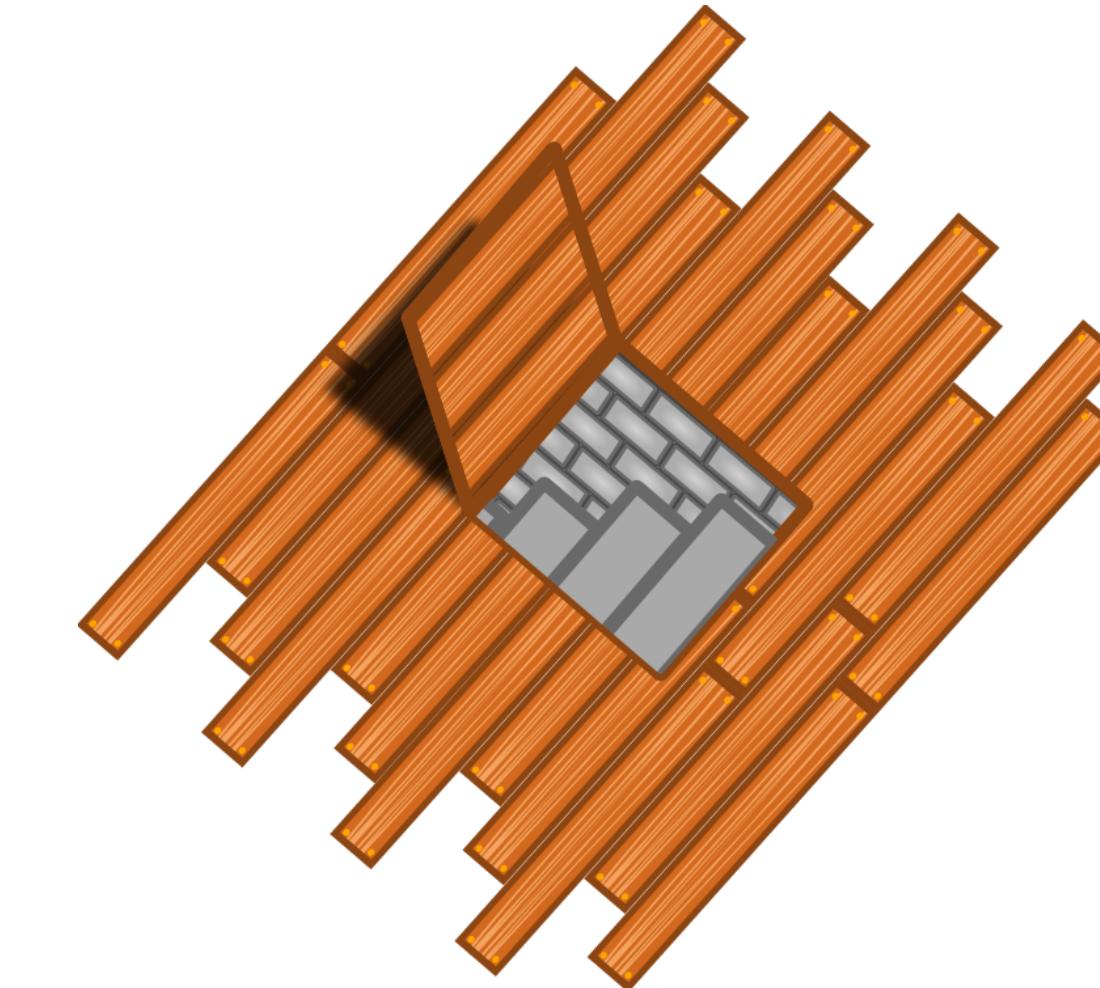
**Are you still
following?**



A Generic Way To Build EUF-CMA Signatures



&



The Hash and Sign Paradigm (Aka Full Domain Hash Signatures)

FDHS Recipe Given a trapdoor one-way function $(KeyGen_{\text{TD}}, F, I)$ over X, Y and a cryptographic hash function $H : \{0,1\}^* \rightarrow X$ build a digital signature scheme as follows

- $KeyGen_{\text{Sig}}(1^n) = KeyGen_{\text{TD}}(1^n) \rightarrow (\text{pk}, \text{sk})$
- $Sign(\text{sk}, m) : \text{compute } \sigma = I(\text{sk}, H(m))$
- $Ver(\text{pk}, m, \sigma) : \text{if } F(\text{pk}, \sigma) = H(m) \text{ return 1, else 0.}$

🤔 Can we use sha256 as hash function for the RSA signature ($\sigma = H(m)^d \pmod N$)?

Theorem

If $(KeyGen_{\text{TD}}, F, I)$ is a one-way trapdoor function and H is a random oracle then FDHS is unforgeable (i.e., FHDS is a chosen message attack secure digital signature scheme)

We will not prove this in the course (if you are curious check out [Theorem 10.17](#))



RSA provides a very versatile trapdoor function: swapping the roles of F and I we get a digital signature scheme!

[more on this in the next lecture]

Digital Signatures Come in Many Flavours

Group Signatures

Attribute-Based Signatures

Threshold Signatures

Key-Homomorphic Signatures

Ring Signatures

Aggregate Signatures

Identity-Based Signatures

[more on these in Lecture 10]

Homomorphic Signatures

Blind Signatures

Functional Signatures

Structure Preserving Signatures

Anonymous Signatures

Proxy Signatures

Redactable Signatures

Multi Signatures

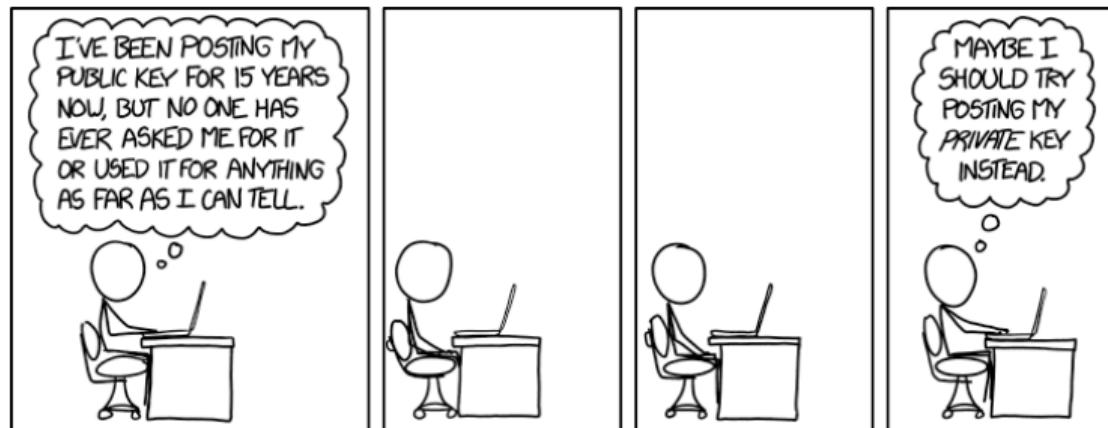
Sequential Signatures

Lecture 7: Public Key Encryption, Homomorphic Encryption

Ivan Oleynikov

Chalmers University

November 26, 2024



Lectures 5 and 6 last week:

- One-Way Trapdoor Functions
- Textbook RSA
- Diffie-Hellman Key Exchange
- Group Theory (Euler, Fermat)
- Homomorphic Signatures

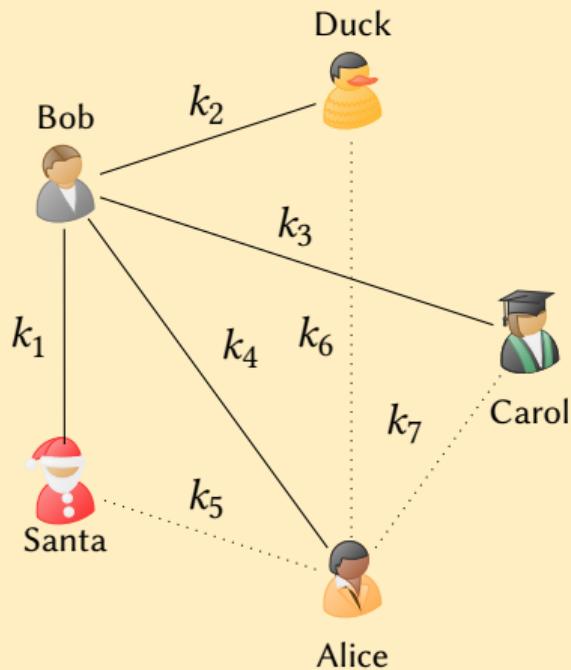
Lecture 7 today:

- Public-Key Encryption
- RSA OAEP
- ElGamal
- IND-CPA and IND-CCA security
- Malleability
- Finding inversions in \mathbb{Z}_N fast
- Linearly Homomorphic Encryption
- Fully Homomorphic Encryption

- Introducing Public-Key Encryption
- Comparing it with One-Way Trapdoor Functions

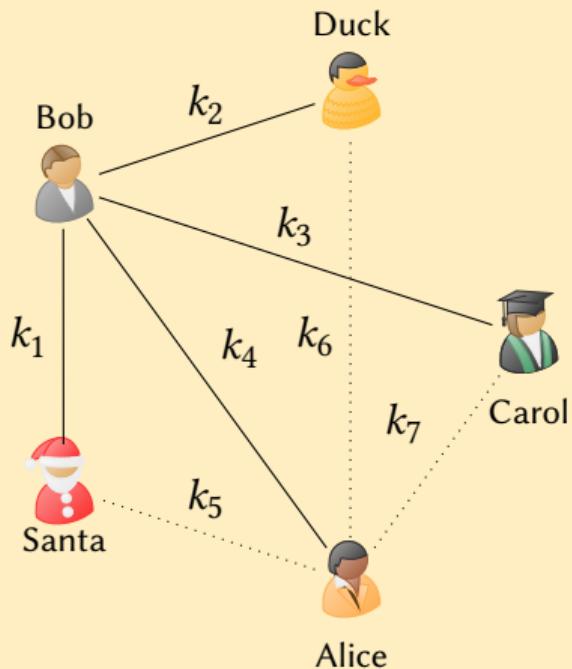
Symmetric Encryption Key Distribution Issue

How many keys do we need so everyone can talk to each other?



Symmetric Encryption Key Distribution Issue

How many keys do we need so everyone can talk to each other?

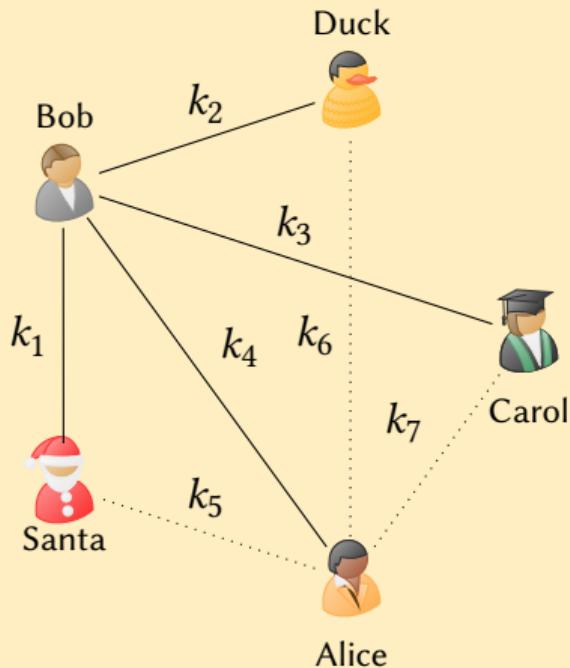


How many keys are needed to connect each pair of 100 users?

- 4 keys for Bob + 3 keys for Alice + 2 keys for ...
- $\frac{n(n-1)}{2}$ keys for n parties

Symmetric Encryption Key Distribution Issue

How many keys do we need so everyone can talk to each other?



How many keys are needed to connect each pair of 100 users?

- 4 keys for Bob + 3 keys for Alice + 2 keys for ...
- $\frac{n(n-1)}{2}$ keys for n parties

Public-Key Encryption, on the other hand, solves this with only n keys!

Public-Key Encryption and One-Way Trapdoor Functions

Public-Key Encryption (PKE)

A Public-Key Encryption is $(\text{KeyGen}, \text{Enc}, \text{Dec})$.
Where KeyGen and Enc are probabilistic poly-time
algorithms (PPT), and Dec is deterministic.

- $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^n)$ is key generation,
produces secret key and public key
- Choose any m (does not have to be random)
- $c \leftarrow \text{Enc}(\text{pk}, m)$ is encryption
- $m' = \text{Dec}(\text{sk}, c)$ is decryption

Notation: $m \in \mathcal{M}, c \in \mathcal{C}$.

Correctness: $m = m'$.

Public-Key Encryption and One-Way Trapdoor Functions

Public-Key Encryption (PKE)

A Public-Key Encryption is $(\text{KeyGen}, \text{Enc}, \text{Dec})$.
Where KeyGen and Enc are probabilistic poly-time algorithms (PPT), and Dec is deterministic.

- $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^n)$ is key generation, produces secret key and public key
- Choose any m (does not have to be random)
- $c \leftarrow \text{Enc}(\text{pk}, m)$ is encryption
- $m' = \text{Dec}(\text{sk}, c)$ is decryption

Notation: $m \in \mathcal{M}, c \in \mathcal{C}$.

Correctness: $m = m'$.

One-Way Trapdoor Function (TDF)

Three algorithms (KeyGen, F, I) where KeyGen is probabilistic, F and I are deterministic.

- $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^n)$
- $x \leftarrow^{\$} \mathcal{X}$ (must be random)
- $y = F(\text{pk}, x)$
- $x' = I(\text{sk}, y)$

Notation: $x, x' \in \mathcal{X}$ and $y \in \mathcal{Y}$.

Correctness: $F(\text{pk}, x') = y$.

(Not guaranteed that $x = x'$.)

Public-Key Encryption and One-Way Trapdoor Functions

Public-Key Encryption (PKE)

A Public-Key Encryption is $(\text{KeyGen}, \text{Enc}, \text{Dec})$.
Where KeyGen and Enc are probabilistic poly-time algorithms (PPT), and Dec is deterministic.

- $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^n)$ is key generation, produces secret key and public key
- Choose any m (does not have to be random)
- $c \leftarrow \text{Enc}(\text{pk}, m)$ is encryption
- $m' = \text{Dec}(\text{sk}, c)$ is decryption

Notation: $m \in \mathcal{M}, c \in \mathcal{C}$.

Correctness: $m = m'$.

One-Way Trapdoor Function (TDF)

Three algorithms (KeyGen, F, I) where KeyGen is probabilistic, F and I are deterministic.

- $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^n)$
- $x \leftarrow^{\$} \mathcal{X}$ (must be random)
- $y = F(\text{pk}, x)$
- $x' = I(\text{sk}, y)$

Notation: $x, x' \in \mathcal{X}$ and $y \in \mathcal{Y}$.

Correctness: $F(\text{pk}, x') = y$.

(Not guaranteed that $x = x'$.)

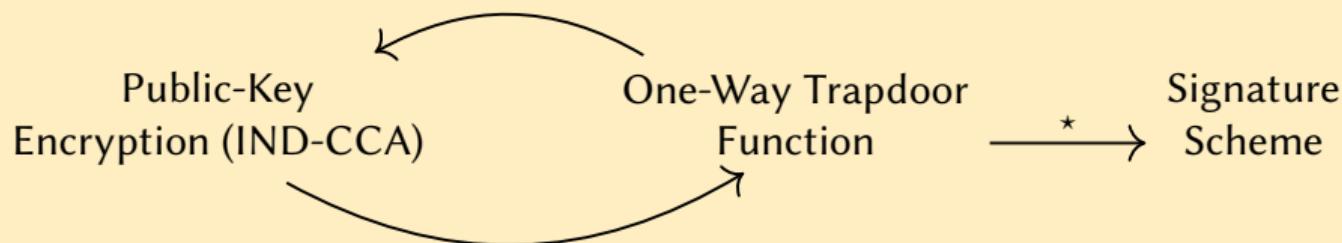
What's the difference?

Why should I care about One-Way Trapdoor Functions?

Uses of TDFs

- One-Way Trapdoor Functions are equivalent to Public-Key Encryption, but have a simpler definition (easier to study).
- They help connect different Public-Key primitives,
- and study the underlying computational assumptions.

Computational Assumptions Hierarchy



* assuming Collision-Resistant Hash functions (Hash-and-Sign, Lecture 6).

One-Way Trapdoor Functions and Public-key Encryption Schemes:

- Textbook RSA
- ElGamal

KeyGen(1^n)

- ① Pick n -bit long primes p and q
- ② $N = pq$
- ③ Pick e such that $\gcd(e, \varphi(N)) = 1$
- ④ $d = e^{-1} \pmod{\varphi(N)}$
- ⑤ $\text{pk} = (N, e)$, $\text{sk} = (N, d)$
- ⑥ return (pk , sk)

Enc $_{(N,e)}(m \in \mathbb{Z}_N^*)$

- ① $c = m^e \pmod{N}$
- ② return c

Dec $_{(N,d)}(c \in \mathbb{Z}_N^*)$

- ① $m = c^d \pmod{N}$
- ② return m

KeyGen(1^n)

- ① Pick n -bit long primes p and q
- ② $N = pq$
- ③ Pick e such that $\gcd(e, \varphi(N)) = 1$
- ④ $d = e^{-1} \pmod{\varphi(N)}$
- ⑤ $\text{pk} = (N, e)$, $\text{sk} = (N, d)$
- ⑥ return (pk , sk)

Enc $_{(N,e)}(m \in \mathbb{Z}_N^*)$

- ① $c = m^e \pmod{N}$
- ② return c

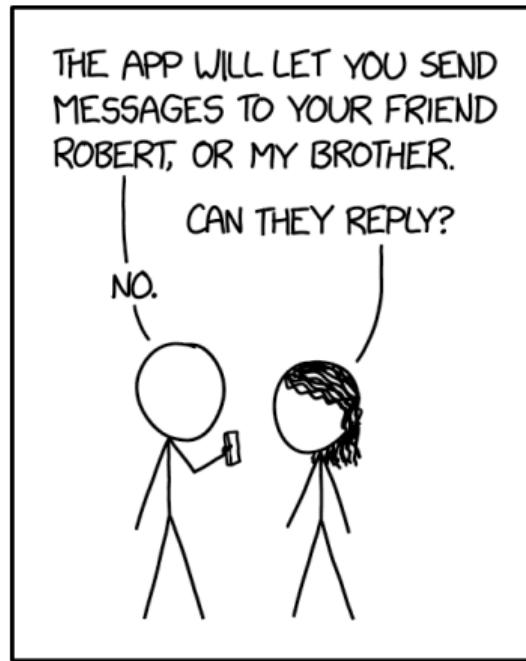
Correctness

$$\begin{aligned} c^d \pmod{N} &= (m^e)^d \pmod{N} \\ &= m \end{aligned}$$

Dec $_{(N,d)}(c \in \mathbb{Z}_N^*)$

- ① $m = c^d \pmod{N}$
- ② return m

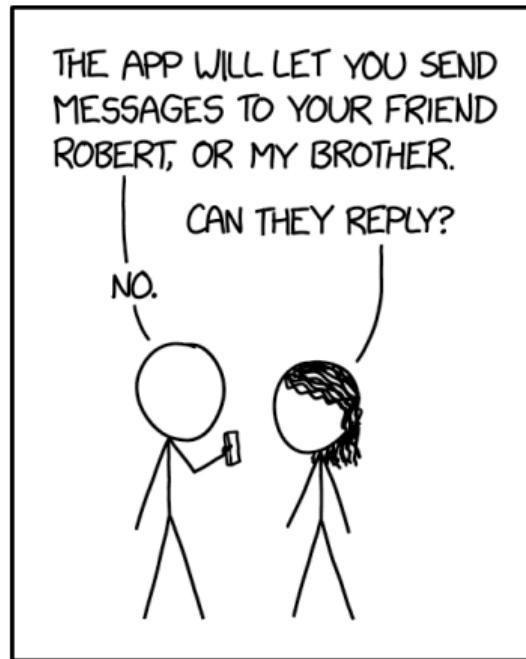
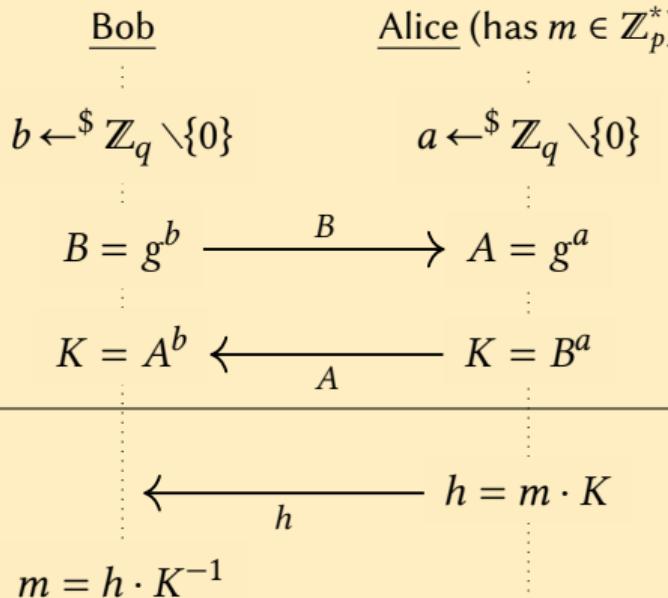
Warmup: Encrypting a Message with Diffie-Hellman



MY NEW SECURE TEXTING APP
ONLY ALLOWS PEOPLE NAMED
ALICE TO SEND MESSAGES
TO PEOPLE NAMED BOB.

Warmup: Encrypting a Message with Diffie-Hellman

Establish a key + transmit a secret message



MY NEW SECURE TEXTING APP
ONLY ALLOWS PEOPLE NAMED
ALICE TO SEND MESSAGES
TO PEOPLE NAMED BOB.

Warmup: Encrypting a Message with Diffie-Hellman

Establish a key + transmit a secret message

$$\begin{array}{ll} \text{Bob} & \text{Alice (has } m \in \mathbb{Z}_p^*) \\ \vdots & \vdots \\ b \xleftarrow{\$} \mathbb{Z}_q \setminus \{0\} & a \xleftarrow{\$} \mathbb{Z}_q \setminus \{0\} \end{array}$$

$$B = g^b \xrightarrow{B} A = g^a$$

$$K = A^b \xleftarrow[A]{} K = B^a$$

$$\xleftarrow[h]{} h = m \cdot K$$

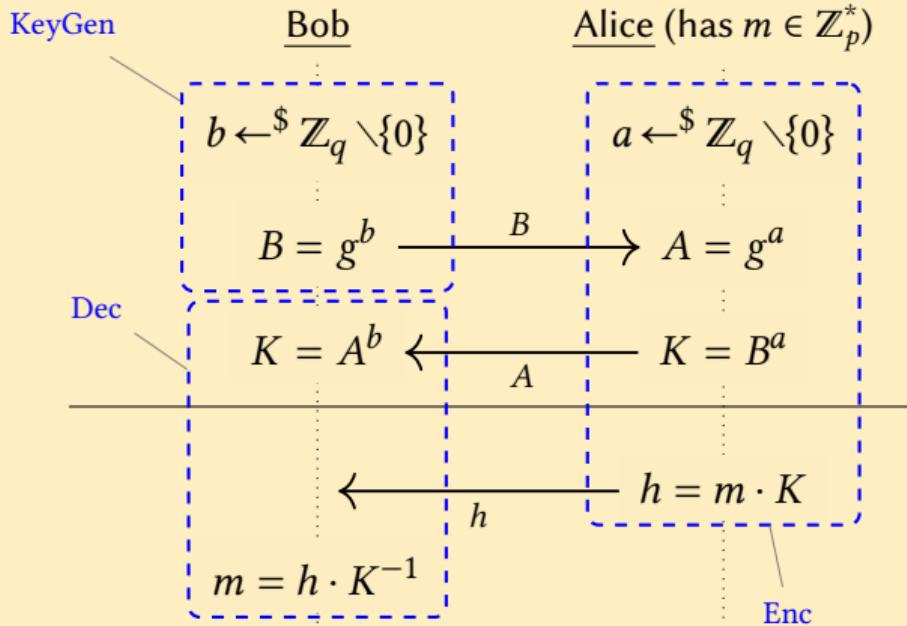
$$m = h \cdot K^{-1}$$

Parameters

- p is a large prime
- g is an element of \mathbb{Z}_p^* which generates a cyclic (multiplicative) subgroup of order q , where q is a prime.

Warmup: Encrypting a Message with Diffie-Hellman

Establish a key + transmit a secret message

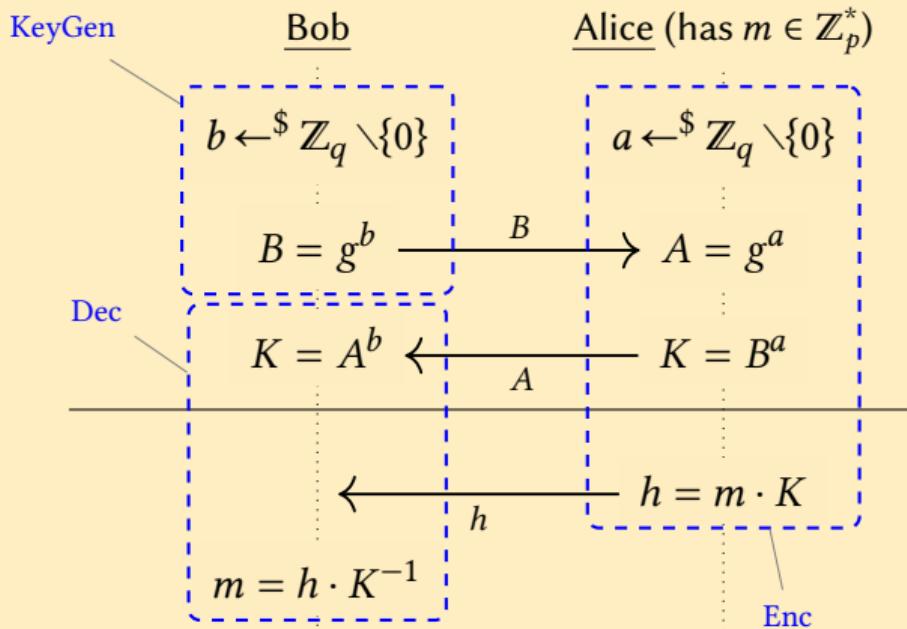


Parameters

- p is a large prime
- g is an element of \mathbb{Z}_p^* which generates a cyclic (multiplicative) subgroup of order q , where q is a prime.

Warmup: Encrypting a Message with Diffie-Hellman

Establish a key + transmit a secret message



Parameters

- p is a large prime
- g is an element of \mathbb{Z}_p^* which generates a cyclic (multiplicative) subgroup of order q , where q is a prime.

In ElGamal terms...

- b is Bob's private key
- B is Bob's public key
- (A, h) is the ciphertext that encrypts Alice's message m for Bob

KeyGen(1^n)

- ① Pick p and $g \in \mathbb{Z}_p^*$ so that g generates a subgroup of order q
- ② $b \leftarrow \$ \mathbb{Z}_q \setminus \{0\}$
- ③ $B = g^b$
- ④ $\text{sk} = b, \text{pk} = B$
- ⑤ return (pk, sk)

Enc_{pk}(m)

- ① $B = \text{pk}$
- ② $a \leftarrow \$ \mathbb{Z}_q \setminus \{0\}$
- ③ $A = g^a$
- ④ $K = B^a$
- ⑤ $h = K \cdot m$
- ⑥ $c = (A, h)$
- ⑦ return c

Dec_{sk}(c)

- ① $b = \text{sk}, (A, h) = c$
- ② $K = A^b$
- ③ $m = h \cdot K^{-1}$
- ④ return m

- Both RSA and ElGamal require computing inverses x^{-1} in \mathbb{Z}_N .
- How to do that efficiently?

Finding Inverses using Euler's Theorem (HA2*)

Applying Euler's Theorem

Let p and q be primes. For any

$$x \in \mathbb{Z}_p^*,$$

$$x^{(p-1)-1} \pmod{p} = x^{-1}.$$

For $M = pq$ and any $x \in \mathbb{Z}_M^*$,

$$x^{(p-1)(q-1)-1} \pmod{M} = x^{-1}.$$

Finding Inverses using Euler's Theorem (HA2*)

Applying Euler's Theorem

Let p and q be primes. For any $x \in \mathbb{Z}_p^*$,

$$x^{(p-1)-1} \bmod p = x^{-1}.$$

For $M = pq$ and any $x \in \mathbb{Z}_M^*$,
 $x^{(p-1)(q-1)-1} \bmod M = x^{-1}$.

Finding Inverses in \mathbb{Z}_M^* : Using Euler

- Suppose $M = p$. Then $x^{-1} = x^{p-2} \bmod M$ can be computed as $\text{binExp}(x, p-2, M)$.
- If $M = pq$, then $x^{-1} = x^{(p-1)(q-1)-1} \bmod M$ is computed as $\text{binExp}(x, (p-1)(q-1)-1, M)$.

Works only if we know the factorization of M .

Finding Inverses using Euler's Theorem (HA2*)

Applying Euler's Theorem

Let p and q be primes. For any $x \in \mathbb{Z}_p^*$,

$$x^{(p-1)-1} \bmod p = x^{-1}.$$

For $M = pq$ and any $x \in \mathbb{Z}_M^*$,
 $x^{(p-1)(q-1)-1} \bmod M = x^{-1}$.

Finding Inverses in \mathbb{Z}_M^* : Using Euler

- Suppose $M = p$. Then $x^{-1} = x^{p-2} \bmod M$ can be computed as $\text{binExp}(x, p-2, M)$.
- If $M = pq$, then $x^{-1} = x^{(p-1)(q-1)-1} \bmod M$ is computed as $\text{binExp}(x, (p-1)(q-1)-1, M)$.

Works only if we know the factorization of M .

Binary Exponentiation Algorithm

$$\text{binExp}(x, i, M) = \begin{cases} 1 & \text{if } i = 0, \\ \text{binExp}(x, i/2, M)^2 \bmod M & \text{else if } i \text{ is even,} \\ \text{binExp}(x, i-1, M) \cdot x \bmod M & \text{else if } i \text{ is odd.} \end{cases}$$

Finding Inverses Using Extended Euclidean Algorithm (HA2*)

Bézout's Identity

Let x and y be positive integers, and $d = \gcd(x, y)$.

Then there exist integers (possibly negative) s and t
such that $xs + yt = d$.

Finding Inverses Using Extended Euclidean Algorithm (HA2*)

Bézout's Identity

Let x and y be positive integers, and $d = \gcd(x, y)$.
Then there exist integers (possibly negative) s and t
such that $xs + yt = d$.

Extended Euclidean Algorithm (EEA)

Given $x, y \in \mathbb{Z}$ find such s, t, d that
 $d = \gcd(x, y)$ and $sx + ty = d$.

EEA(x, y)

if $x = 0$ **then**

return $(y, 0, 1)$

$(d, s', t') := \text{EEA}(y \bmod x, x)$

$s := t' - \lfloor y/x \rfloor \cdot s'$

$t := s'$

return (d, s, t)

Finding Inverses Using Extended Euclidean Algorithm (HA2*)

Bézout's Identity

Let x and y be positive integers, and $d = \gcd(x, y)$. Then there exist integers (possibly negative) s and t such that $xs + yt = d$.

Finding Inverses in \mathbb{Z}_M^* : Using Bézout

Given M and $x \in \mathbb{Z}_M^*$, the $x^{-1} \pmod M$ can be computed via

- ① $(1, s, t) = \text{EEA}(x, M)$.
- ② **return** $(s \pmod M)$.

Proof: $xs + Mt \pmod M = 1$, therefore $xs + Mt \pmod M = 1$, so $xs \pmod M = 1$. And $s \pmod M = x^{-1}$.

Works for any M .

Extended Euclidean Algorithm (EEA)

Given $x, y \in \mathbb{Z}$ find such s, t, d that $d = \gcd(x, y)$ and $sx + ty = d$.

EEA(x, y)

if $x = 0$ **then**

return $(y, 0, 1)$

$(d, s', t') := \text{EEA}(y \pmod x, x)$

$s := t' - \lfloor y/x \rfloor \cdot s'$

$t := s'$

return (d, s, t)

Security Properties:

- One-Way Trapdoor Functions
- Public-Key Encryption
 - IND-CPA: INDistinguishability under Chosen Plaintext Attack
 - IND-CCA: INDistinguishability under Chosen Ciphertext Attack
- Malleability

Security of One-Way Trapdoor Functions

TDF Game

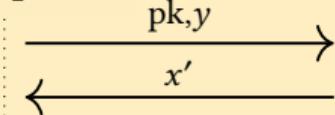
Challenger

\mathcal{A}

$(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^n)$

$x \xleftarrow{\$} \mathcal{X}$

$y = F(\text{pk}, x)$



$F(\text{pk}, x) = F(\text{pk}, x')?$

\mathcal{A} wins if $F(\text{pk}, x) = F(\text{pk}, x')$.

Definition

(KeyGen, F, I) is a secure TDF if for any PPT \mathcal{A}

$$\Pr[\mathcal{A} \text{ wins TDF Game}] = \text{negl}(n).$$

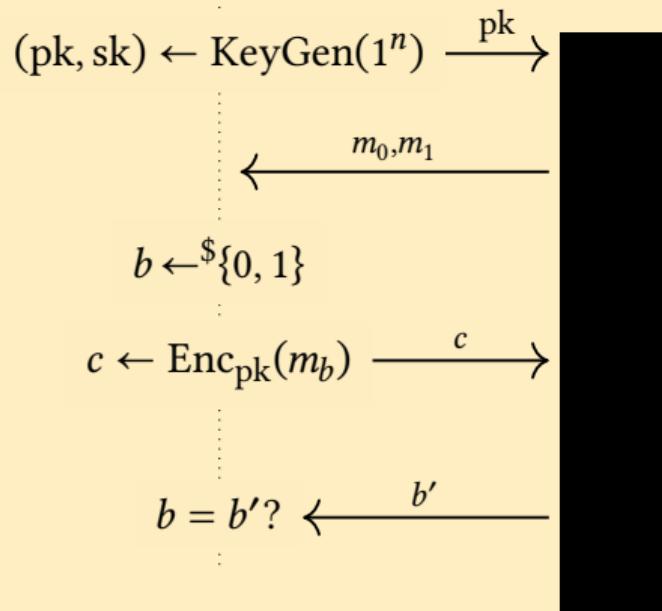
Remarks

- TDF Game is not using sk .
- Condition $F(\text{pk}, x) = F(\text{pk}, x')$ is not the same as $x = x'$ due to collisions.
- Textbook RSA is believed to be a secure TDF.

IND-CPA: INDistinguishability under Chosen Plaintext Attack

IND-CPA Game

Challenger



Definition

$(\text{KeyGen}, \text{Enc}, \text{Dec})$ is Indistinguishable under Chosen-Plaintext Attack (IND-CPA) if for any PPT \mathcal{A}

$$\Pr[b = b'] = 1/2 + \text{negl}(n).$$

Remarks

- Adversary can encrypt any messages of its choice using pk .
- Deterministic encryption (like Textbook RSA) can't be IND-CPA secure.
- ElGamal is IND-CPA secure (under DDH), Textbook RSA is not.

Malleability: Bug or Feature?

Definition

A PKE Scheme is malleable if one can modify a ciphertext and predict how that impacts the encrypted message (knowing pk, but not the message or sk).

For example: for any ciphertext c that decrypts to m , produce a ciphertext c' that will decrypt to $2 \cdot m$ (without knowing m or sk).

ElGamal is malleable

Given: pk and c such that $\text{Enc}_{pk}(m) = c$.

Compute: $(A, h) = c$ and
 $c' = (A, 2 \cdot h \bmod p)$.

We know that $\text{Dec}_{sk}(c') = (2 \cdot m) \bmod p$.

Proof: $A^{-b} \cdot 2 \cdot (B^a m) = 2m$.

Textbook RSA is malleable

Given: pk and c such that $\text{Enc}_{pk}(m) = c$.

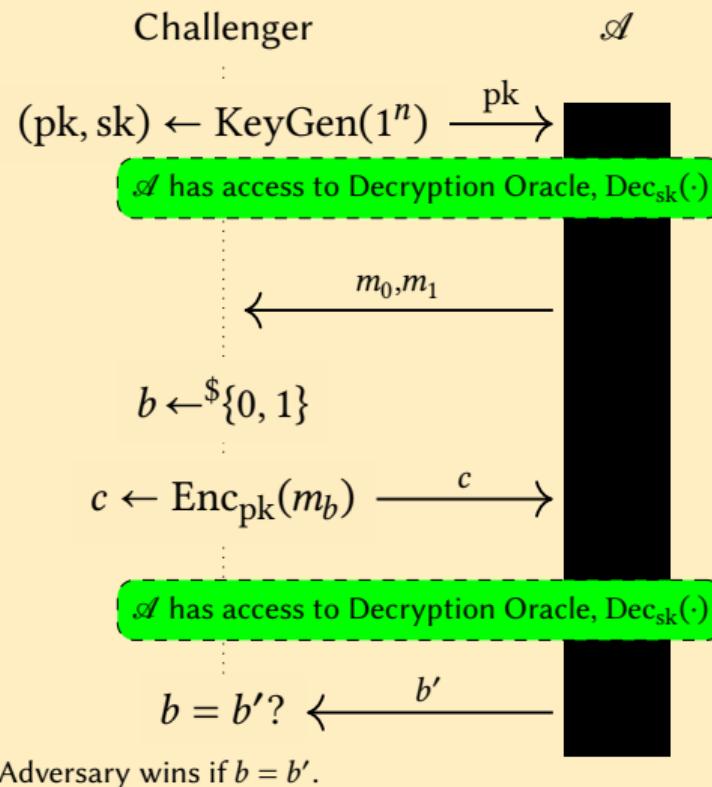
Compute: $e = pk$ and $c' = 2^e \cdot c \bmod N$.

We know that $\text{Dec}_{sk}(c') = (2 \cdot m) \bmod p$.

Proof: $(2^e \cdot m^e)^d = 2m$.

IND-CCA: INDistinguishability under Chosen Ciphertext Attack

IND-CCA Game



Definition

$(\text{KeyGen}, \text{Enc}, \text{Dec})$ is IND-CCA secure if for any PPT \mathcal{A}

$$\Pr[b = b'] = 1/2 + \text{negl}(n),$$

and \mathcal{A} never requested $\text{Dec}_{\text{sk}}(c)$, where c is the challenge it got.

Remarks

- Adversary can encrypt any messages of its choice using pk.
- Malleable encryption can't be IND-CCA secure.
- ElGamal is malleable, thus not IND-CCA secure.

RSA Optimal Asymmetric Encryption Padding, OAEP

Implementing IND-CCA secure requires both non-malleability, and randomized encryption. OAEP padding ensures both¹.

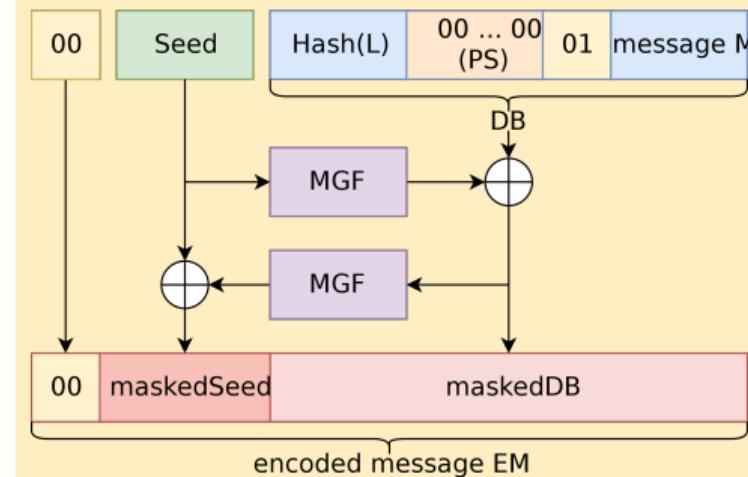
Legend

- L = public label
- Seed = random seed
- Mask Generation Function (MGF) is a special pseudorandom function, similar to hash functions

RSA-OAEP is proven to be IND-CCA secure (in the Random Oracle model)
(Textbook RSA is neither IND-CPA nor IND-CCA secure, but merely a TDF.)

¹we provide this without proof

Encoding



ElGamal and RSA: Properties Summary

	Textbook RSA	ElGamal	RSA OAEP
Deterministic?	yes	no	no
Secure TDF?	yes*	no	no
Malleable?	yes	yes	no
IND-CPA?	no	yes*	yes*
IND-CCA?	no	no	yes*

* under the corresponding computational assumptions

Homomorphic encryption: meaningful computations on private, encrypted data.

- Linearly homomorphic: may only “add” or “subtract”.
- Fully homomorphic: may “add”, “subtract” or “multiply”.

Definition

LHE is a public-key encryption (KeyGen , Enc , Dec) with additional algorithm \oplus_{pk} :

$$\text{Enc}_{\text{pk}}(a) \oplus_{\text{pk}} \text{Enc}_{\text{pk}}(b) = \text{Enc}_{\text{pk}}(a \star b).$$

The meaning of arithmetic operations \star on plaintexts depends on the concrete scheme. Usually, it's modular arithmetic or group operations.

- The homomorphic operation \oplus_{pk} does not need sk or message.
- LHE is always malleable, so it can't be IND-CCA secure.
- LHE allows limited (but occasionally useful) computations on encrypted data.

Linearly Homomorphic Encryption: Familiar Examples

ElGamal

- Let (B_1, c_1) encrypt m_1 with key A
- Let (B_2, c_2) encrypt m_2 with key A
- Suppose Alice is decrypting
 $(B_1, c_1) \oplus (B_2, c_2) =$
 $(B_1 \cdot B_2 \bmod p, c_1 \cdot c_2 \bmod p)$ with her private key a :

$$K = (B_1 B_2)^a$$

$$= B_1^a \cdot B_2^a$$

$$m = c_1 c_2 \cdot K^{-1} \bmod p$$

$$= (c_1 B_1^{-a})(c_2 B_2^{-a}) \bmod p$$

$$= m_1 m_2 \bmod p$$

Linearly Homomorphic Encryption: Familiar Examples

ElGamal

- Let (B_1, c_1) encrypt m_1 with key A
- Let (B_2, c_2) encrypt m_2 with key A
- Suppose Alice is decrypting
 $(B_1, c_1) \oplus (B_2, c_2) =$
 $(B_1 \cdot B_2 \bmod p, c_1 \cdot c_2 \bmod p)$ with her private key a :

$$\begin{aligned} K &= (B_1 B_2)^a \\ &= B_1^a \cdot B_2^a \end{aligned}$$

$$\begin{aligned} m &= c_1 c_2 \cdot K^{-1} \bmod p \\ &= (c_1 B_1^{-a})(c_2 B_2^{-a}) \bmod p \\ &= m_1 m_2 \bmod p \end{aligned}$$

Textbook RSA

- Let m_1^e and m_2^e be RSA ciphertexts, encrypted for Alice whose private key is (d, N)
- Let's see what happens if Alice decrypts $m_1^e \oplus m_2^e = m_1^e m_2^e$:

$$\begin{aligned} m &= (m_1^e m_2^e)^d \bmod N \\ &= (m_1 \cdot m_2)^{ed} \bmod N \\ &= m_1 m_2 \bmod N \end{aligned}$$

Definition

FHE is a Public-Key Encryption Scheme with two extra algorithms:

- $\text{Enc}_{\text{pk}}(a) \oplus_{\text{pk}} \text{Enc}_{\text{pk}}(b) = \text{Enc}_{\text{pk}}(a * b)$
(same as LHE),
- $\text{Enc}_{\text{pk}}(a) \odot_{\text{pk}} \text{Enc}_{\text{pk}}(b) = \text{Enc}_{\text{pk}}(\underbrace{a * a * \dots * a}_{b \text{ times}})$

(Note that both a and b are encrypted and private.)

- FHE are much more versatile than LHE,
- but are more complex and require heavier computations.
- FHE are malleable, can't be IND-CCA secure.

Applications:

- Cloud that can process people's private data without being able to see it.
- Multi-Party Computation where parties jointly perform operations on their data without anyone of them being able to see the data they're operating on.

Homomorphic Linear Operations

For a group (\mathbb{G}, \star) , the \star is called “linear operation” or “group element addition”.

- For ElGamal, the group is (\mathbb{Z}_p^*, \star) with $m_1 \star m_2 = m_1 \cdot m_2 \pmod p$.
- For RSA, the group is (\mathbb{Z}_N^*, \star) with $m_1 \star m_2 = m_1 \cdot m_2 \pmod N$.

Therefore, these are called Linearly Homomorphic.

Homomorphic Multiplication

In terms of group (\mathbb{G}, \star) , “multiplication” is taking $h \in \mathbb{G}$ and $k \in \{0, 1 \dots |\mathbb{G}|\}$, and computing $h^k = \underbrace{h \star h \star \dots \star h}_{k \text{ times}}$.

Fully Homomorphic schemes allow the following two operations:

“Addition” : Given $\text{Enc}_{\text{pk}}(m_1)$ and $\text{Enc}_{\text{pk}}(m_2)$, compute $\text{Enc}_{\text{pk}}(m_1 \star m_2)$.

“Multiplication” : Given $\text{Enc}_{\text{pk}}(m_1)$ and $\text{Enc}_{\text{pk}}(m_2)$, compute $\text{Enc}_{\text{pk}}(m_1^{m_2})$.

- ① Public-Key primitives involve a pair of keys: public pk and secret sk. Different keys enable parties to do different things.
- ② Public-Key Encryption (PKE) uses pk to encrypt and sk to decrypt.
 - IND-CPA: knowing pk is not enough to distinguish encrypted messages.
 - IND-CCA: knowing pk and seeing decryptions of some messages is not sufficient to distinguish (other) messages.
 - Deterministic encryption (like Textbook RSA) can't be IND-CPA.
 - Malleable encryption (like ElGamal) can't be IND-CCA.
- ③ One-Way Trapdoor Functions (TDF) help cryptographers study connections between Public-Key primitives.
 - TDF can be built from IND-CPA PKE (not shown today).
 - IND-CPA and IND-CCA PKE can be built from TDF (not shown today).
- ④ Homomorphic Encryption is PKE where one can perform operations on encrypted values without knowing sk.
 - Homomorphic Encryption is always malleable.
 - Can be used to outsource computations to an untrusted party.



<https://forms.office.com/e/K9sDktPcUg>

- The RSA-OAEP Encoding Diagram was taken from Wikipedia² (CC License).
- People icons in diagrams were taken from tikzpeople LATEX package³
- Font used is Libertinus Sans.
- XKCD comics are from <https://xkcd.com/1553>,
<https://xkcd.com/2691> (CC License).

²https://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding

³<https://www.ctan.org/pkg/tikzpeople>

CRYPTOGRAPHY

The Signal Protocol

Lecturer Hanna

Lecture Agenda

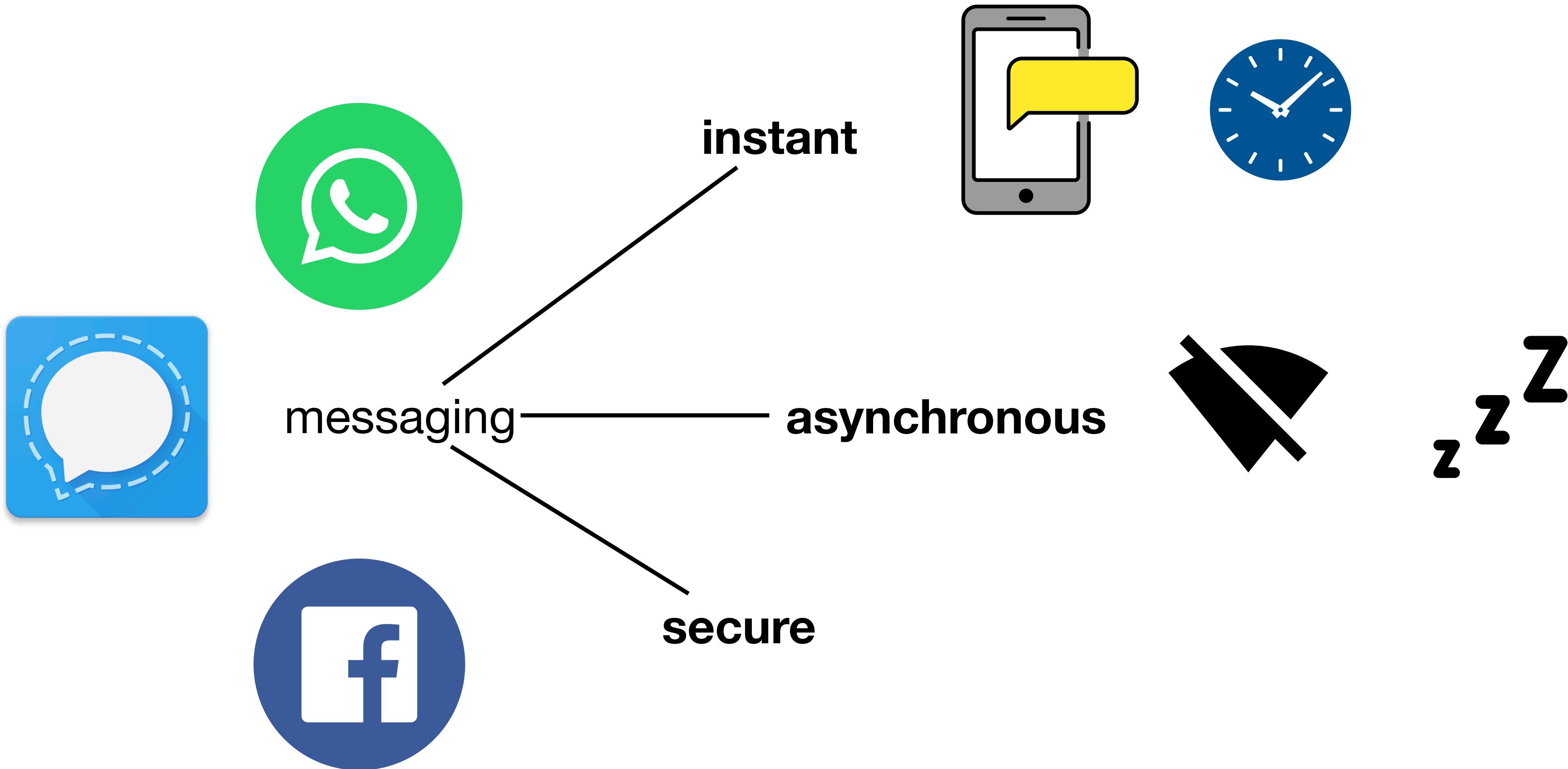
Secure Instant Messaging

- **Security Notions**
 - Backward/Forward Security
- **Tools**
 - AEAD
 - X3DH Protocol
- **The Signal Protocol**
 - Double Ratchet Mechanism
 - Session Hijacking Attack

Recap

- Module 2
- Group Theory

Signal: Privacy That Fits Your Pockets



Cryptographic Building Blocks

5.2. Recommended cryptographic algorithms

Taken from [here](#)

The following choices are recommended for instantiating the cryptographic functions from [Section 3.1](#):

- ***GENERATE_DH()***: This function is recommended to generate a key pair based on the [Curve25519](#) or [Curve448](#) elliptic curves [7].
- ***DH(dh_pair, dh_pub)***: This function is recommended to return the output from the X25519 or X448 function as defined in [7]. There is no need to check for invalid public keys.
- ***KDF_RK(rk, dh_out)***: This function is recommended to be implemented using [HKDF \[3\]](#) with [SHA-256](#) or [SHA-512](#) [8], using *rk* as HKDF *salt*, *dh_out* as HKDF *input key material*, and an application-specific byte sequence as HKDF *info*. The *info* value should be chosen to be distinct from other uses of HKDF in the application.
- ***KDF_CK(ck)***: [HMAC \[2\]](#) with [SHA-256](#) or [SHA-512](#) [8] is recommended, using *ck* as the HMAC key and using separate constants as input (e.g. a single byte 0x01 as input to produce the message key, and a single byte 0x02 as input to produce the next chain key).
- ***ENCRYPT(mk, plaintext, associated_data)***: This function is recommended to be implemented with an [AEAD](#) encryption scheme based on either SIV or a composition of [CBC with HMAC](#) [5], [9]. These schemes provide some misuse-resistance in case a key is mistakenly used multiple times. A concrete recommendation based on

tiny variation of HMAC

Lecture Agenda

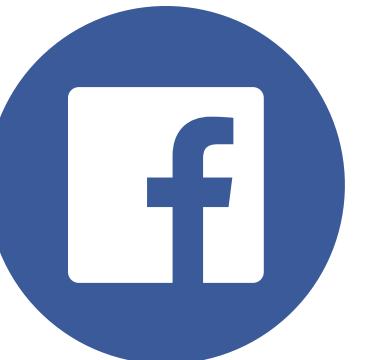
Secure Instant Messaging

- **Security Notions**
 - Backward/Forward Security
- **Tools**
 - AEAD
 - X3DH Protocol
- **The Signal Protocol**
 - Double Ratchet Mechanism
 - Session Hijacking Attack

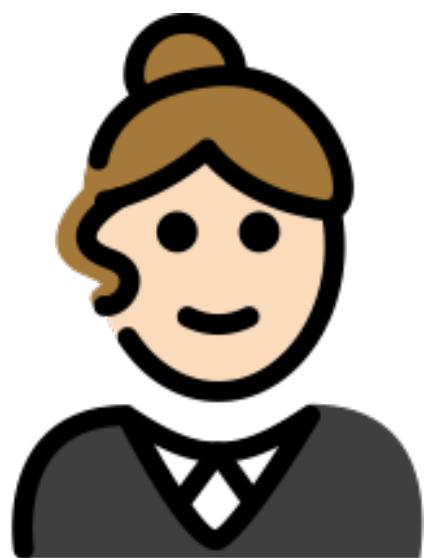
Recap

- Module 2
- Group Theory

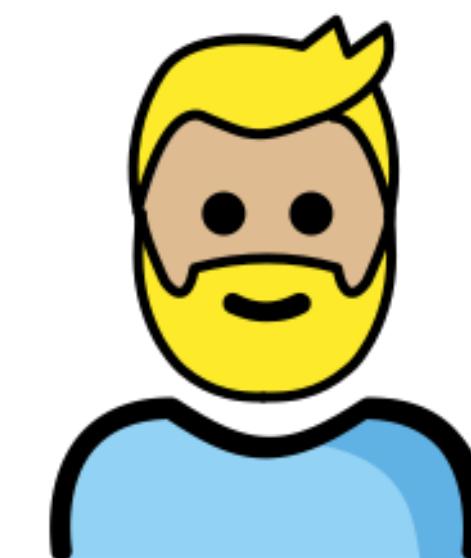
Signal: Privacy That Fits Your Pockets



secure messaging



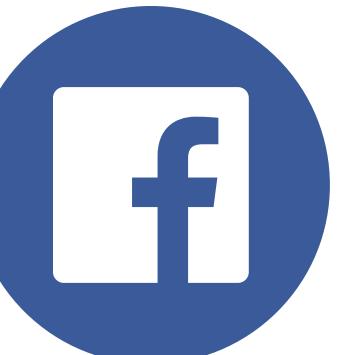
Alice



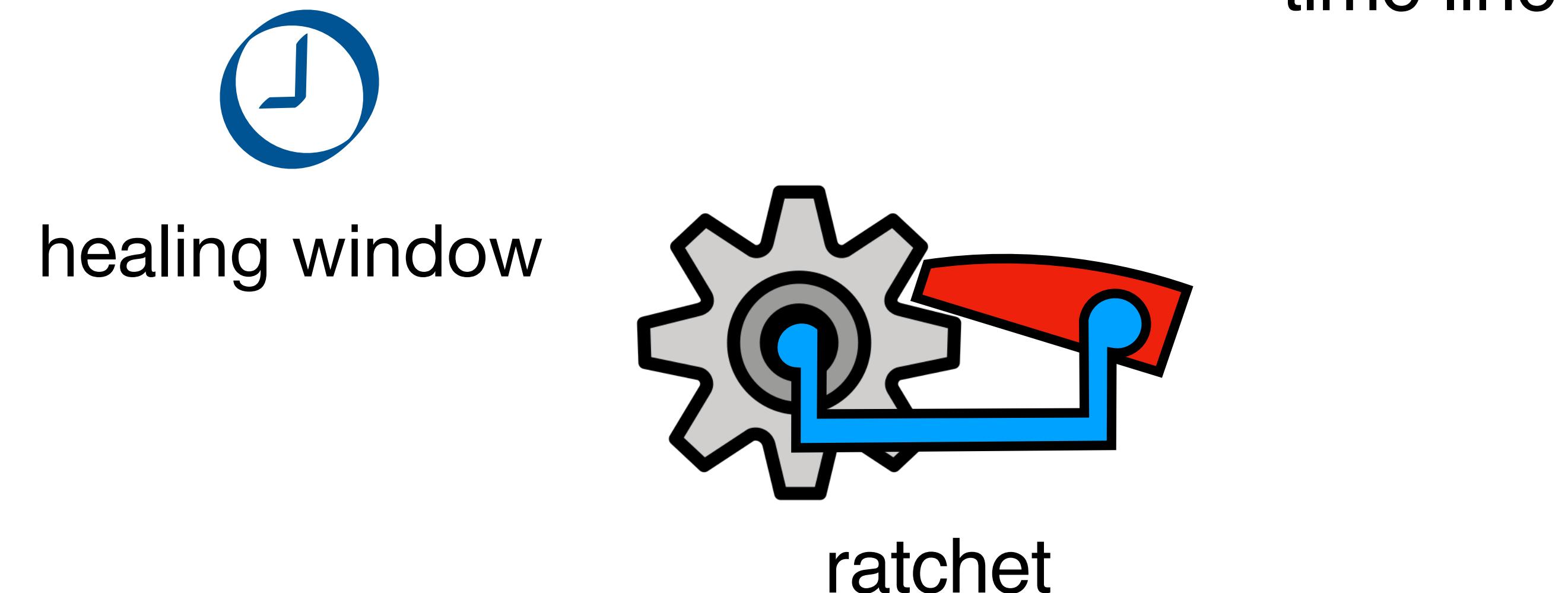
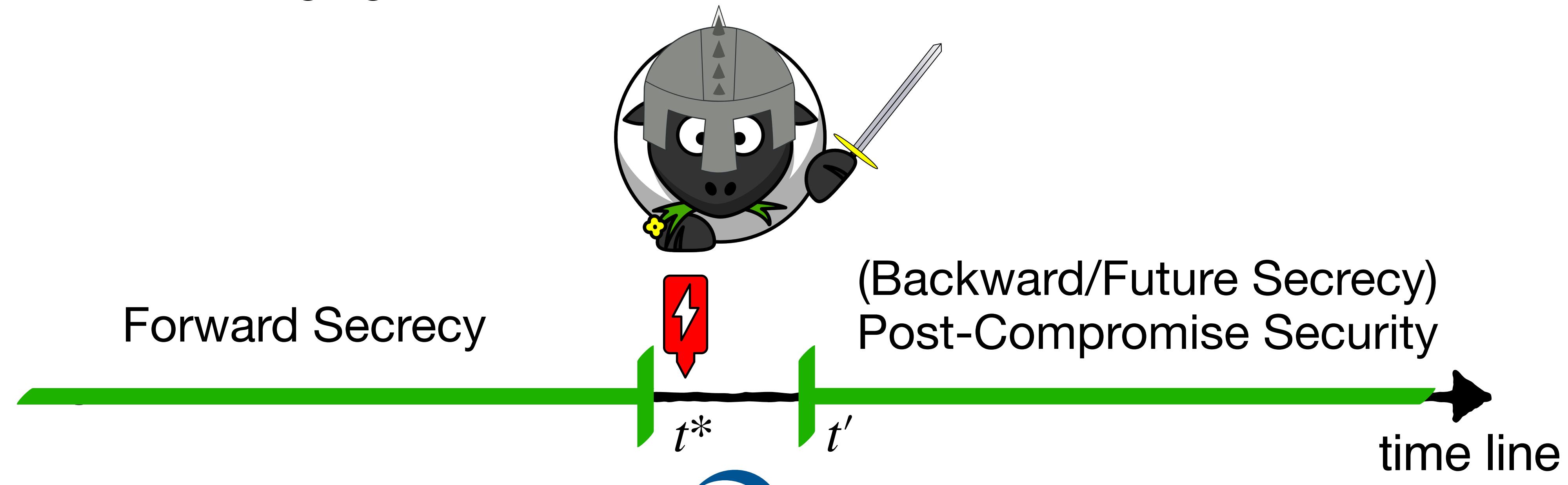
Bob

- end-to-end encryption
- interlocutor authentication
- message integrity

Signal: Privacy That Fits Your Pockets



secure messaging



Lecture Agenda

Secure Instant Messaging

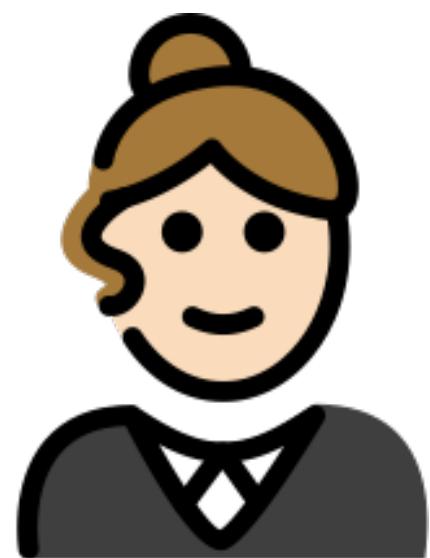
- **Security Notions**
 - Backward/Forward Security
- **Tools**
 - AEAD
 - X3DH Protocol
- **The Signal Protocol**
 - Double Ratchet Mechanism
 - Session Hijacking Attack

Recap

- Module 2
- Group Theory

AEAD: Authenticated Encryption With Associated Data

secure messaging

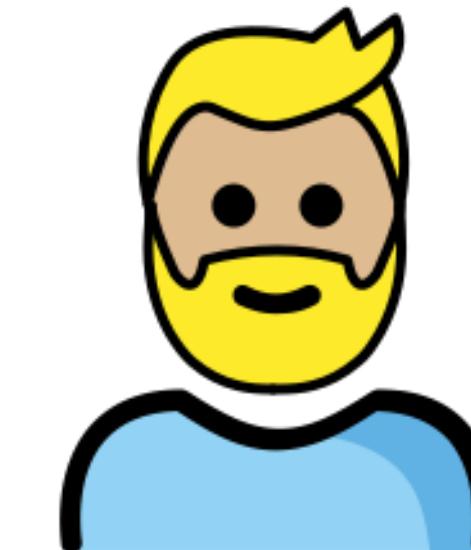


Alice



AEAD

Authenticated
Encryption with
Associated
Data



Bob

How?

$$\text{AEAD}_k[msg, AD] = (c, AD, t)$$

- end-to-end encryption
- interlocutor authentication
- message integrity

Lecture Agenda

Secure Instant Messaging

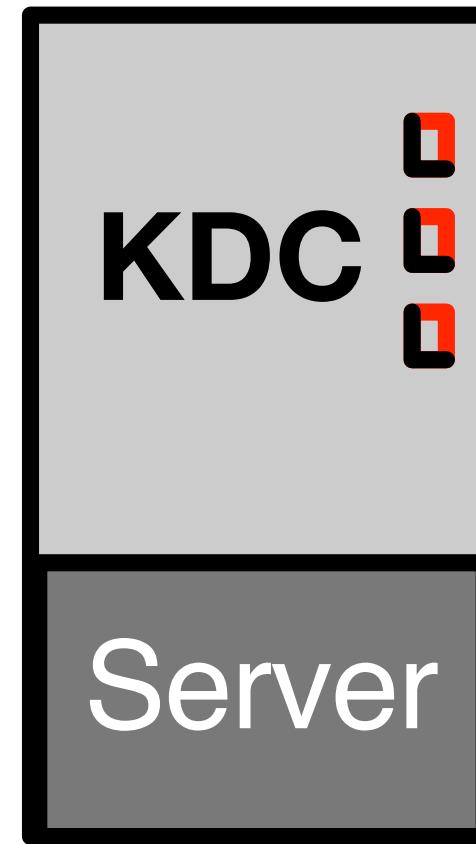
- **Security Notions**
 - Backward/Forward Security
- **Tools**
 - AEAD
 - X3DH Protocol
- **The Signal Protocol**
 - Double Ratchet Mechanism
 - Session Hijacking Attack

• Asynchronous
Key exchange

Recap

- Module 2
- Group Theory

X3DH: Registration Phase



Bob's key bundle

$$\begin{pmatrix} pk_B \\ mtpk_B \\ otpk_B^{(1)} \\ otpk_B^{(2)} \\ \vdots \\ otpk_B^{(N)} \\ sgn_B \end{pmatrix}$$

$\text{Sign}(sk_B, mtpk_B) \rightarrow$

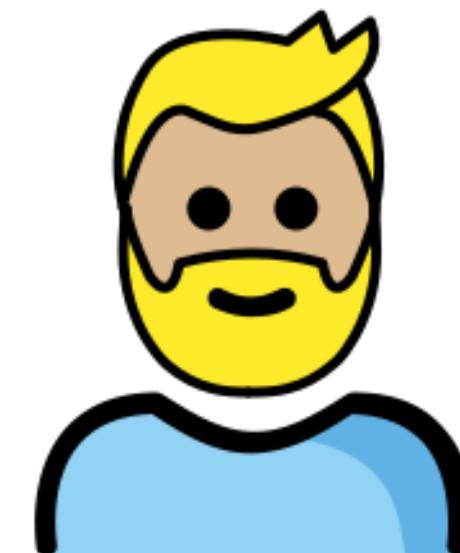
DH keys

$$sk = x \leftarrow \$ - \mathbb{Z}_q^* \\ pk = g^x \in \mathbb{G}$$

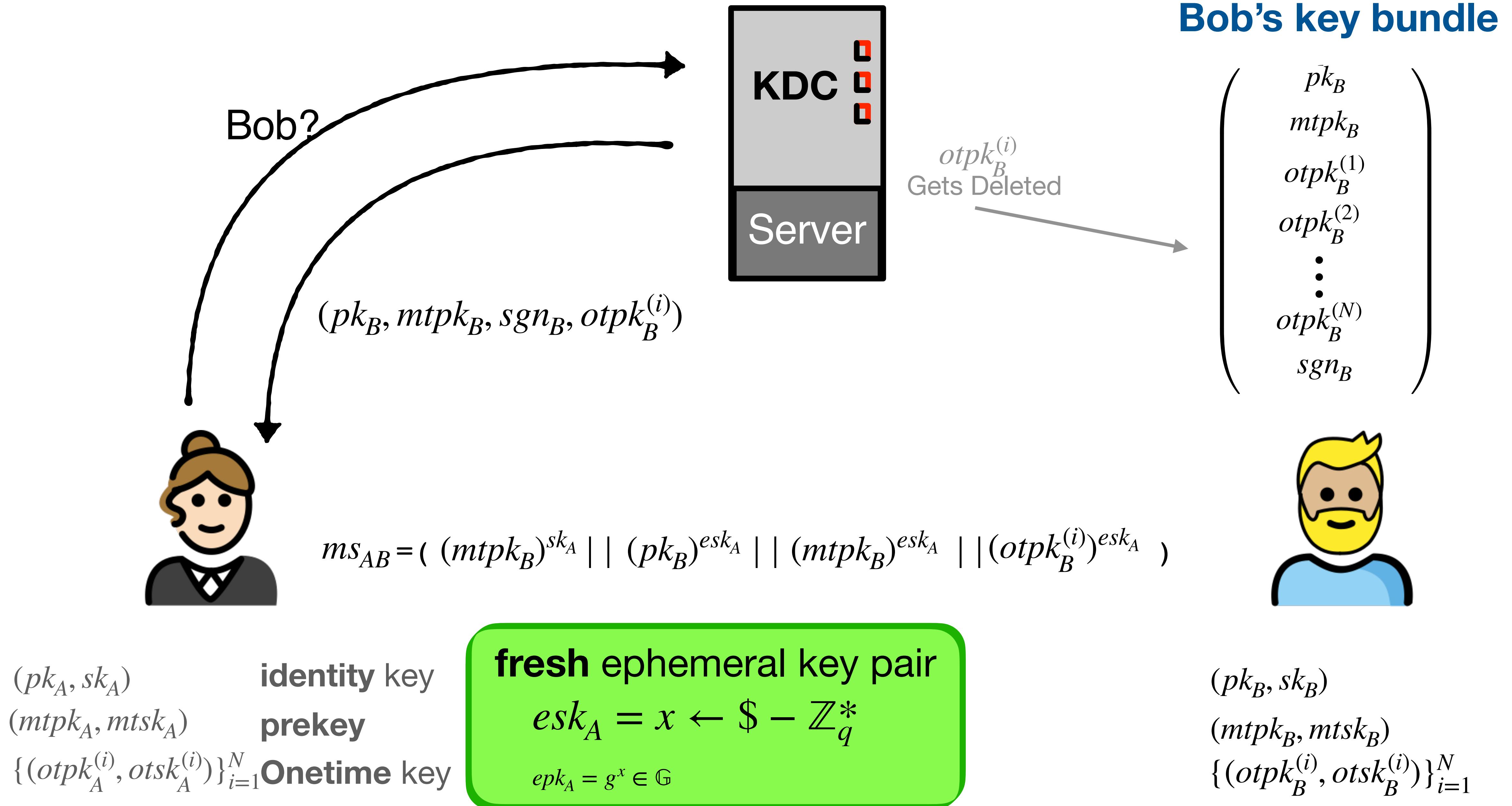
one long term identity key pair (pk_B, sk_B)

one medium term **prekey** pair $(mtpk_B, mtsk_B)$

multiple **one-time** key pairs $\{(otpk_B^{(i)}, otsk_B^{(i)})\}_{i=1}^N$



X3DH: Session Setup Phase



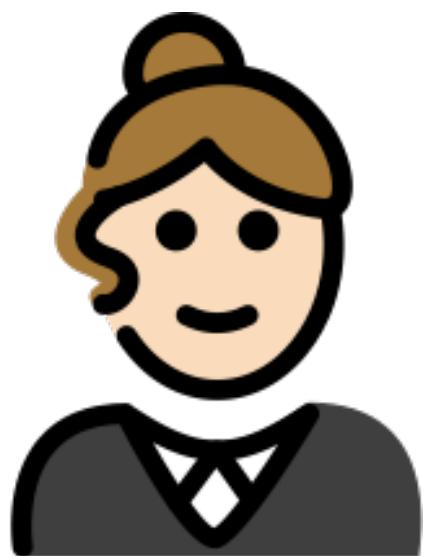
X3DH: Session Setup Phase

(i, AD)

$(pk_A, epk_A) = \text{Decode}(AD)$

$ms_{AB} = ((pk_A)^{mtsk_B} || (epk_A)^{sk_B} || (epk_A)^{mtsk_B} || (epk_A)^{otsk_B^{(i)}})$

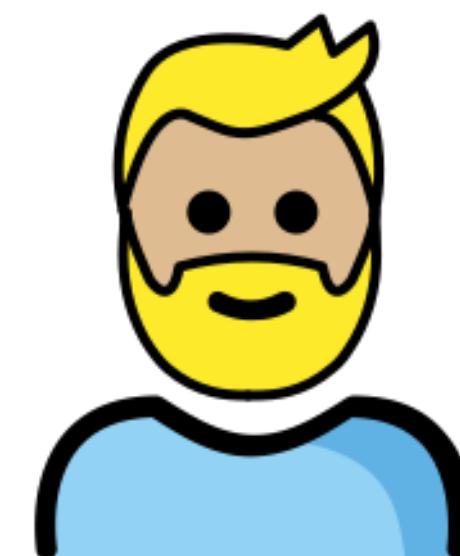
$ms = \text{KDF}(ms_{AB})$



$AD = \text{Encode}((pk_A)) || \text{Encode}((epk_A))$

$ms = \text{KDF}(ms_{AB})$ (Similar to a HMAC)

$ms_{AB} = ((mtpk_B)^{sk_A} || (pk_B)^{esk_A} || (mtpk_B)^{esk_A} || (otpk_B^{(i)})^{esk_A})$



(pk_A, sk_A)

identity key

$(mtpk_A, mtsk_A)$

Mid-term prekey

$\{(otpk_A^{(i)}, otsk_A^{(i)})\}_{i=1}^N$

Onetime key

(pk_B, sk_B)

$(mtpk_B, mtsk_B)$

$\{(otpk_B^{(i)}, otsk_B^{(i)})\}_{i=1}^N$

Lecture Agenda

Secure Instant Messaging

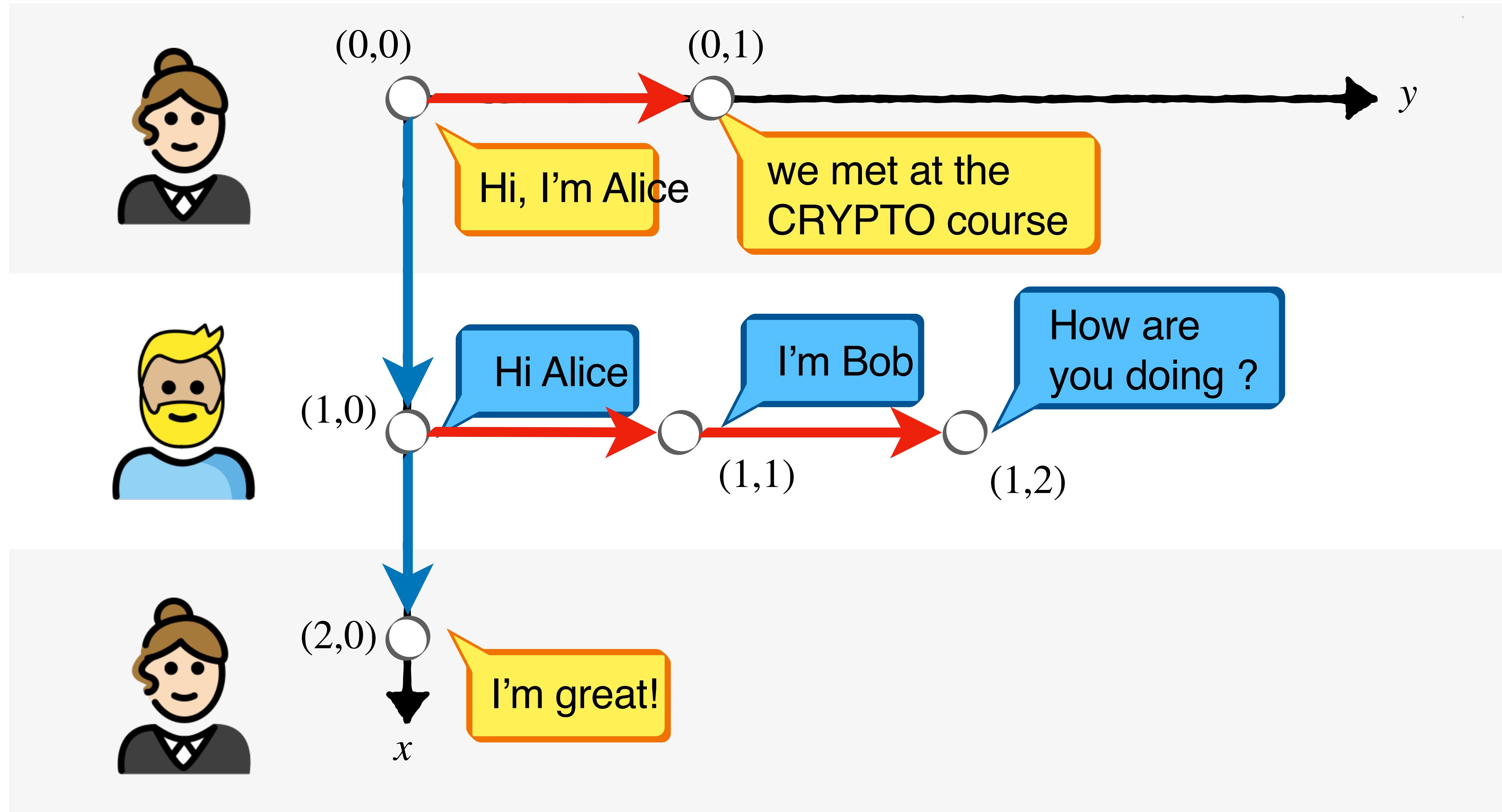
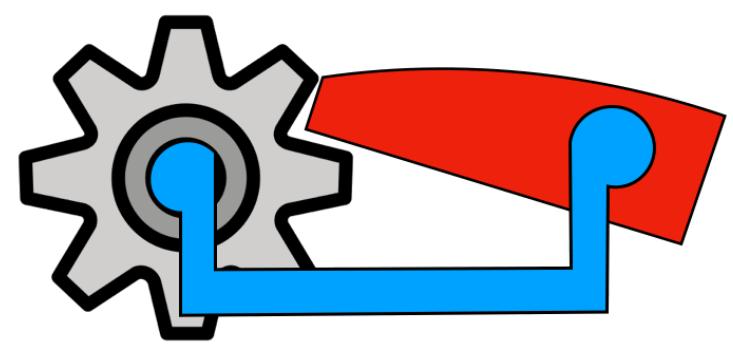
- **Security Notions**
 - Backward/Forward Security
- **Tools**
 - AEAD
 - X3DH Protocol
- **The Signal Protocol**
 - Double Ratchet Mechanism
 - Session Hijacking Attack

Recap

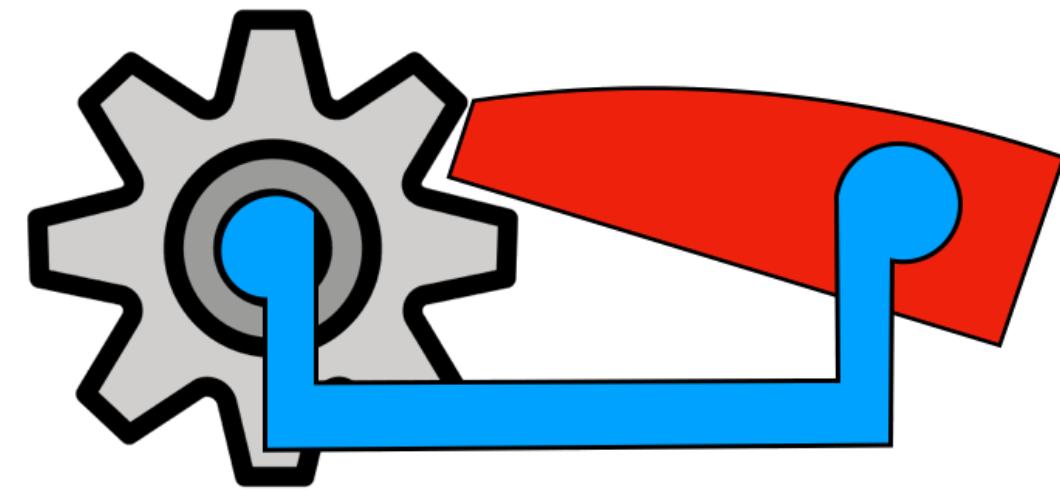
- Module 2
- Group Theory

- Forward Security
- Backward Security

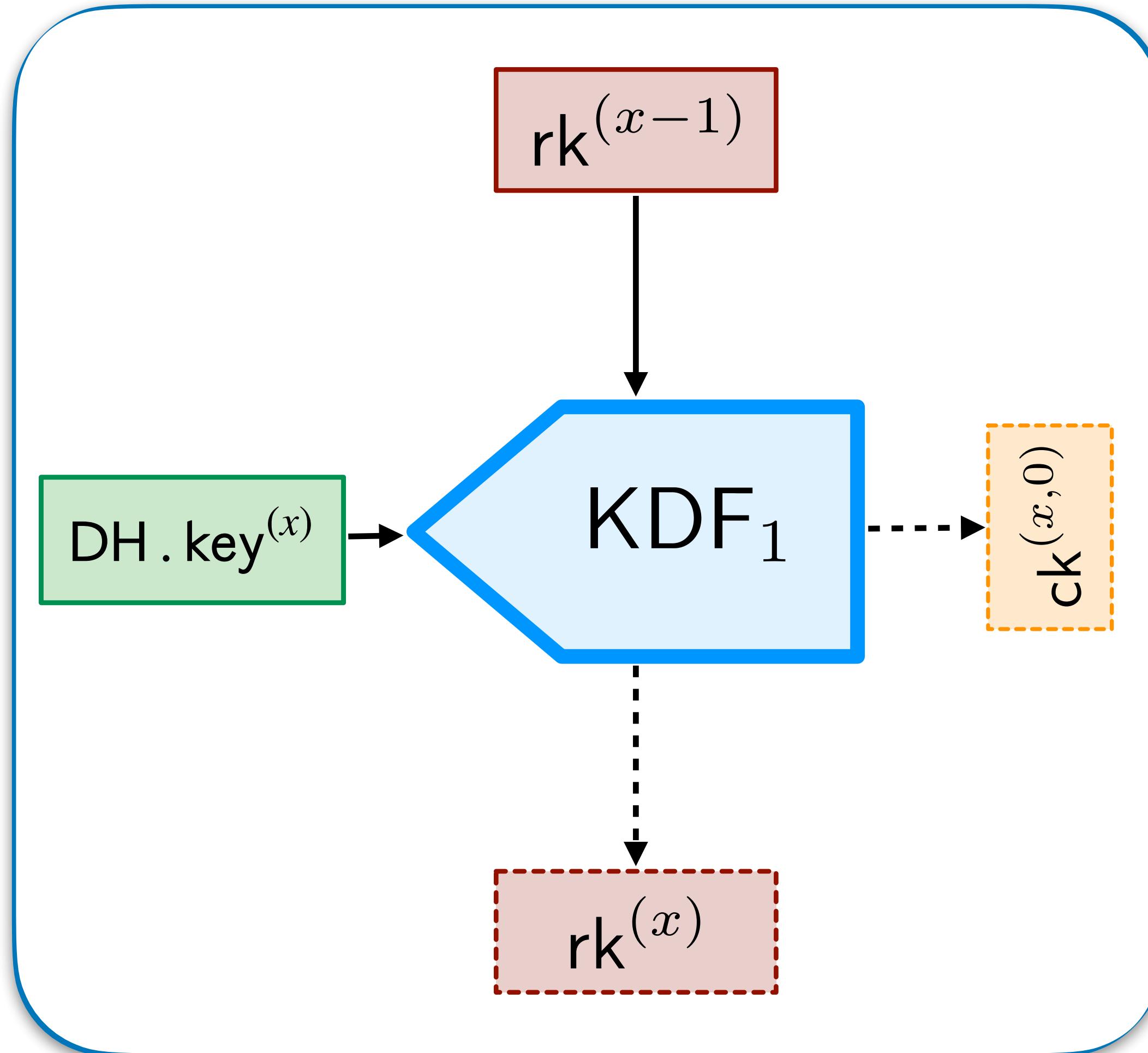
The Time Line of Asynchronous Messaging



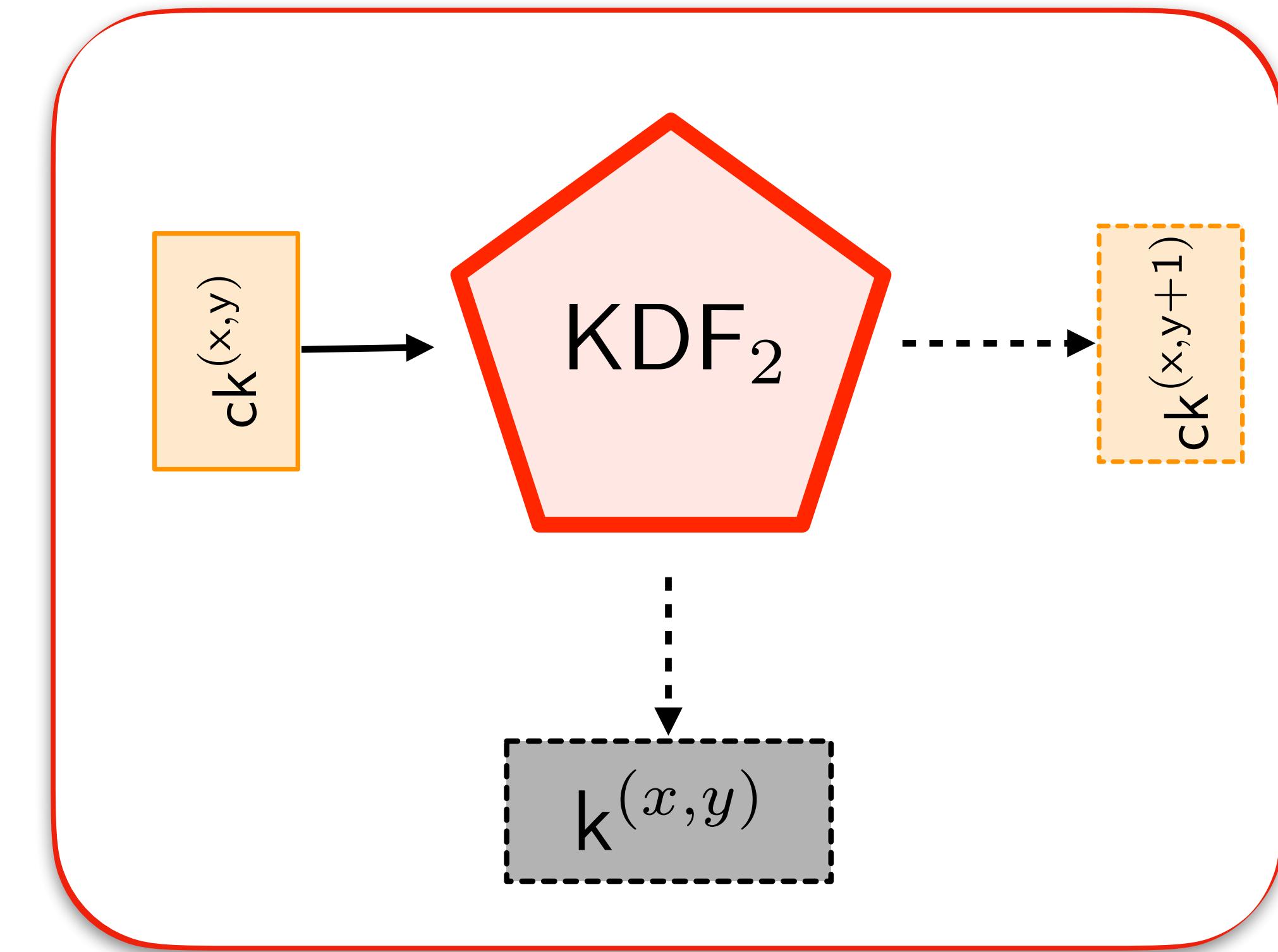
Asymmetric & Symmetric Ratcheting



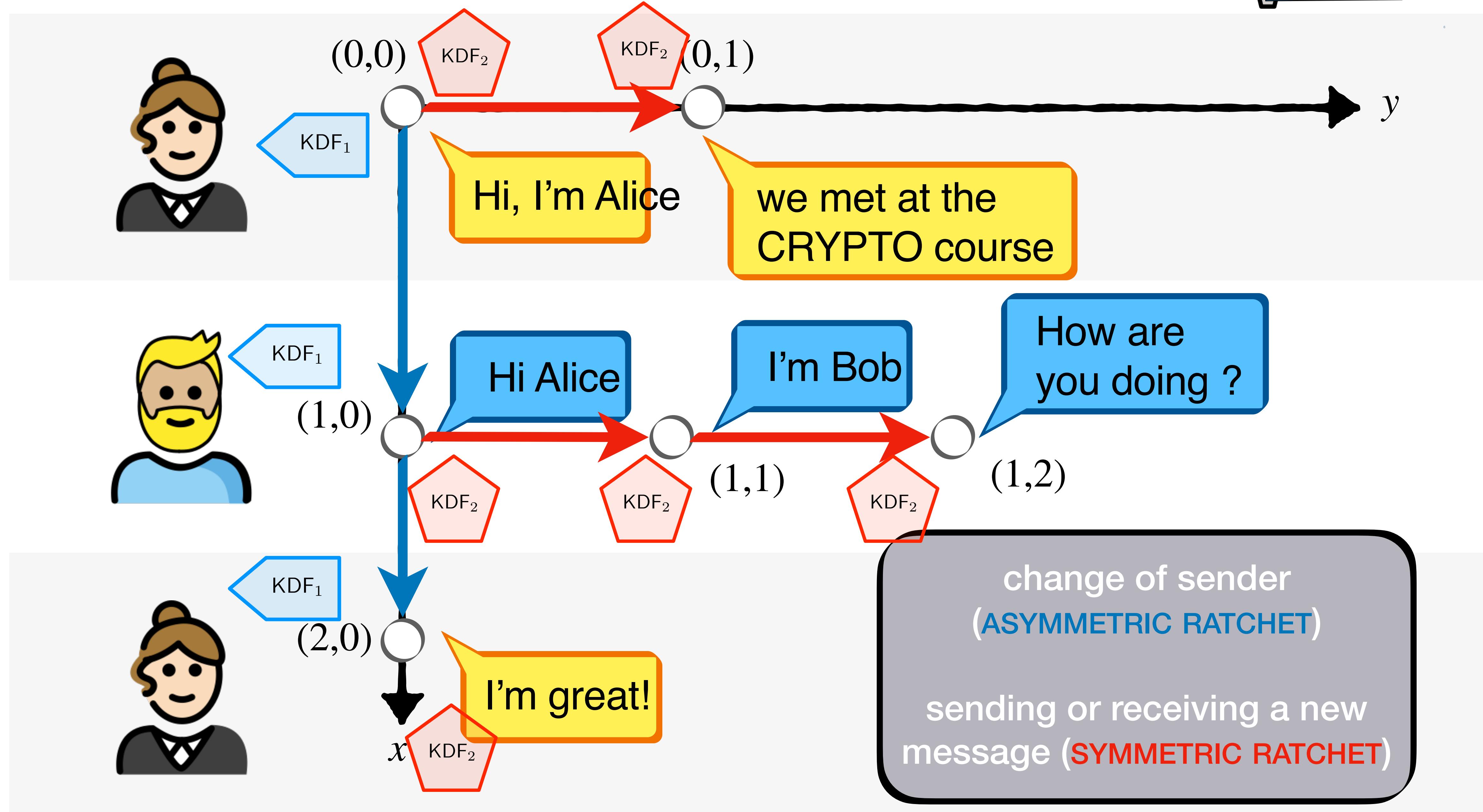
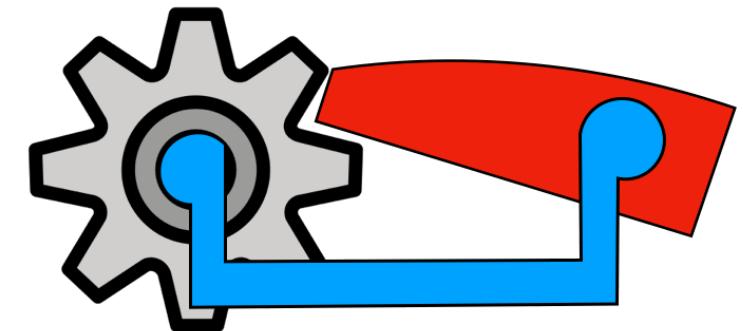
ASYMMETRIC RATCHET



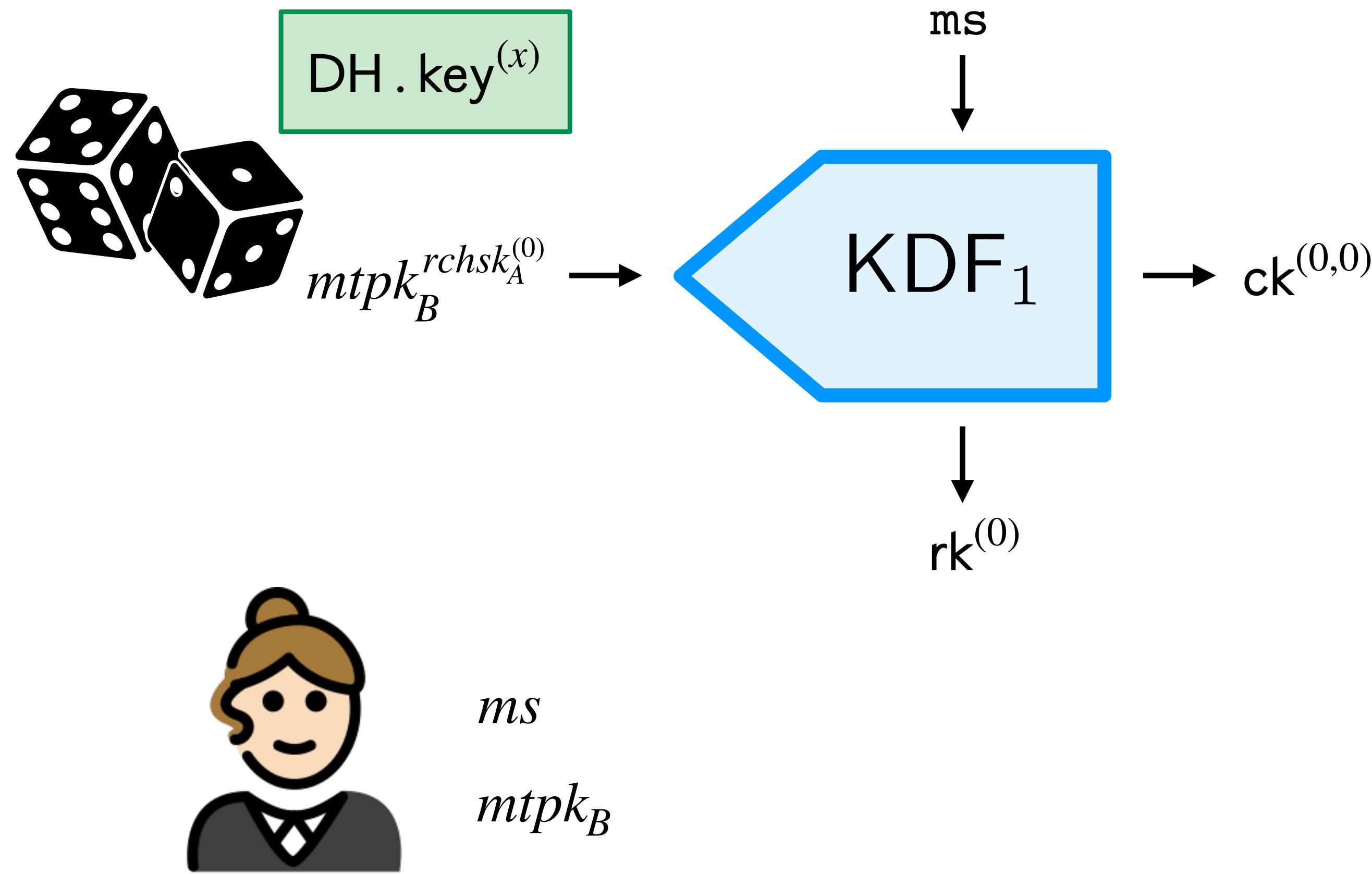
SYMMETRIC RATCHET



The Time Line of Asynchronous Messaging



Signal: Asymmetric Ratchet (First Round)

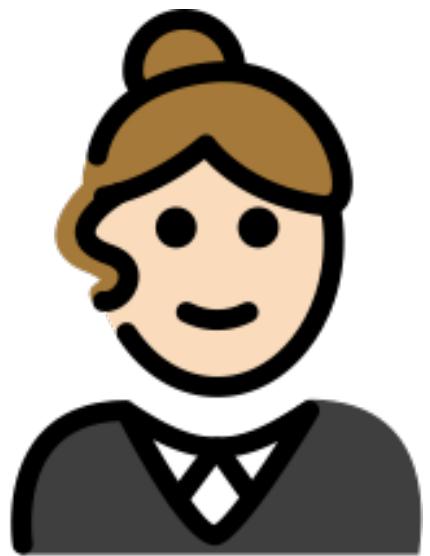
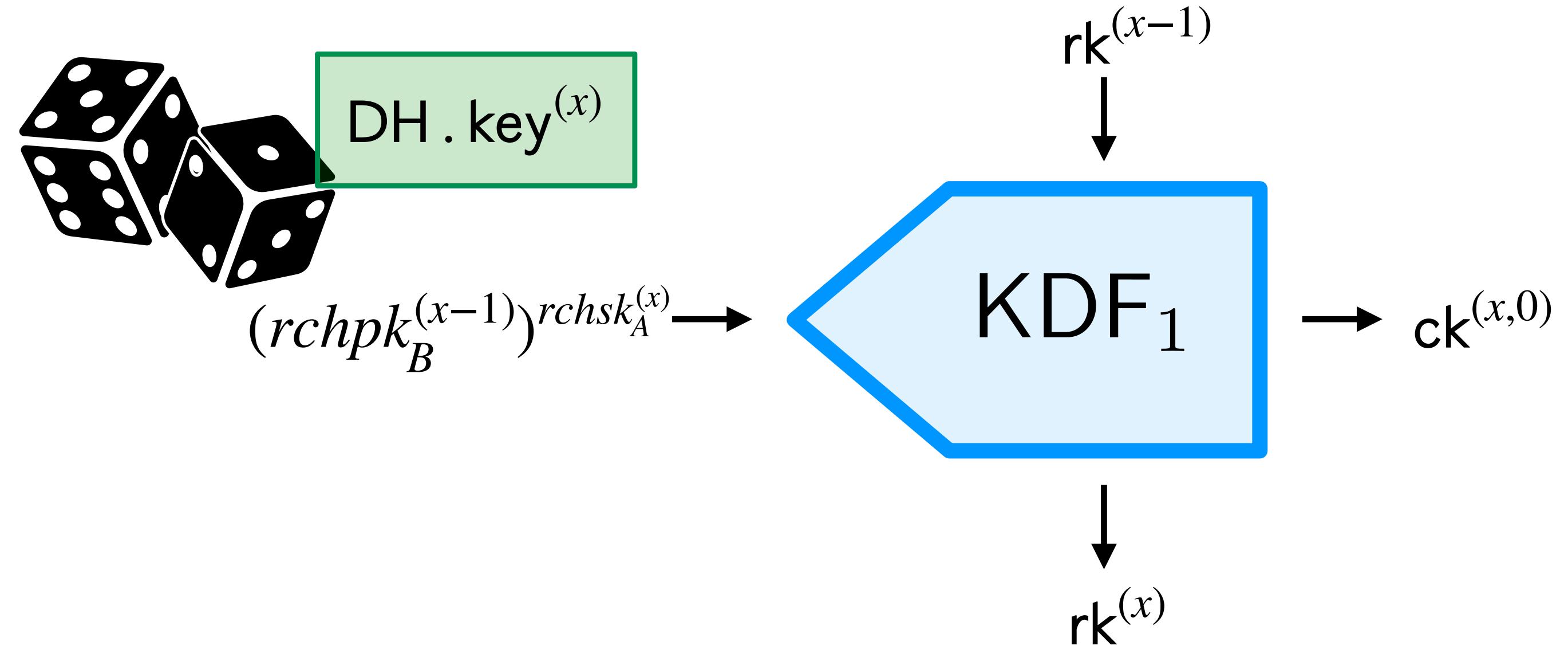


fresh ratchet key pair

$$rchsk_A^{(0)} = z \leftarrow \$ - \mathbb{Z}_q$$

$$rchpk_A^{(0)} = g^z \in \mathbb{G}$$

Signal: Asymmetric Ratchet (Round X)



rk^(x-1)

rchpk_B^(x-1)

fresh ratchet key pair

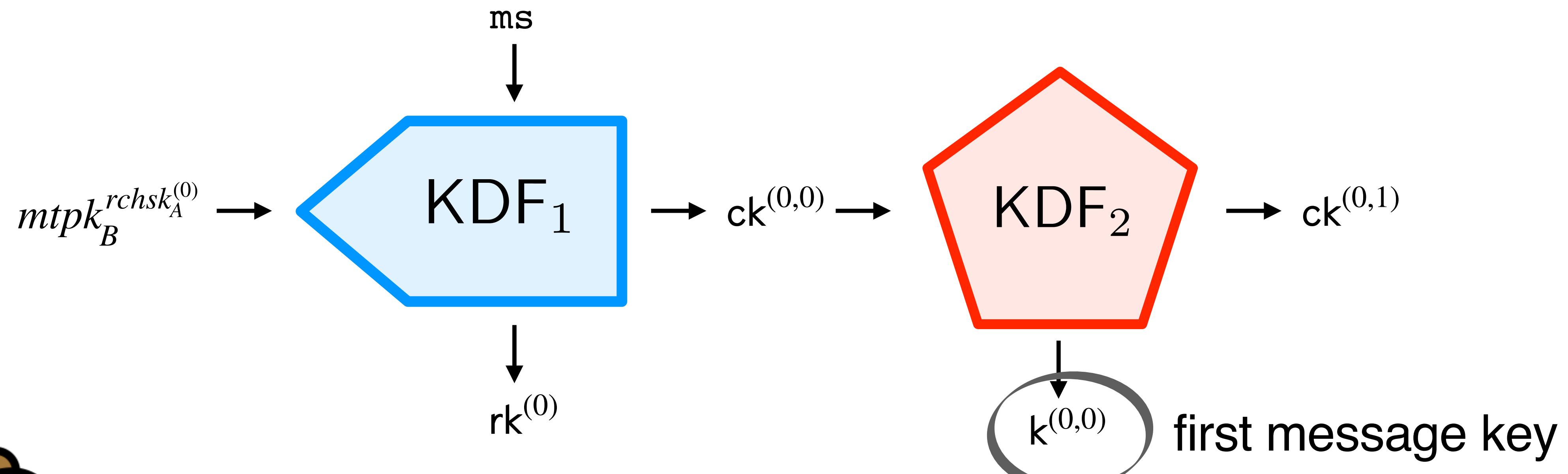
$$rchsks_A^{(x)} = z \leftarrow \$ - \mathbb{Z}_q$$

$$rchpk_A^{(x)} = g^z \in \mathbb{G}$$

$$\begin{aligned} \text{DH . key}^{(x)} &= (rchpk_B^{(x-1)})^{rchsks_A^{(x)}} \\ &= (rchpk_A^{(x)})^{rchsks_B^{(x-1)}} \end{aligned}$$

Signal: Asymmetric Ratchet

... And Symmetric Ratchet

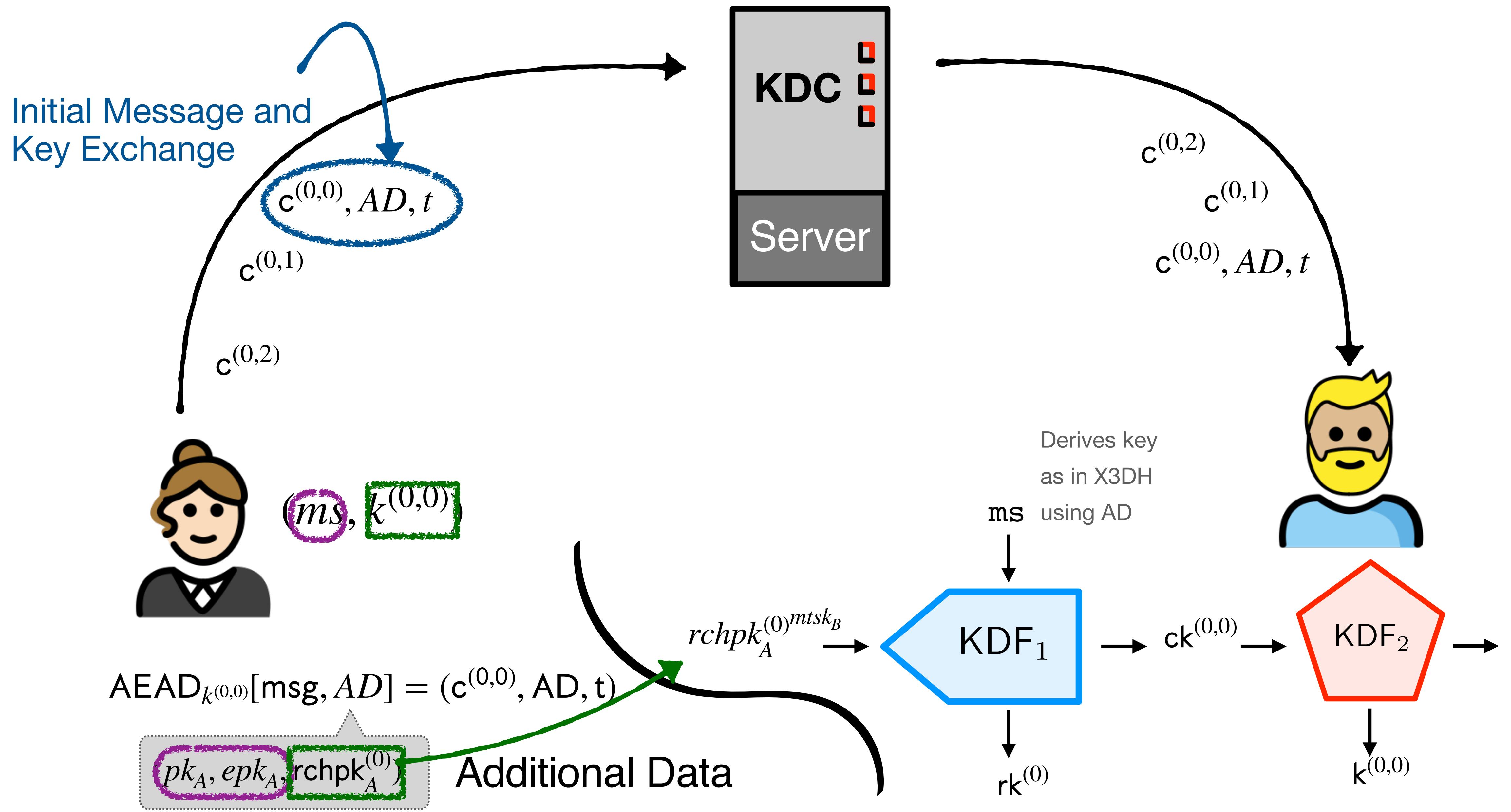


fresh ratchet key pair

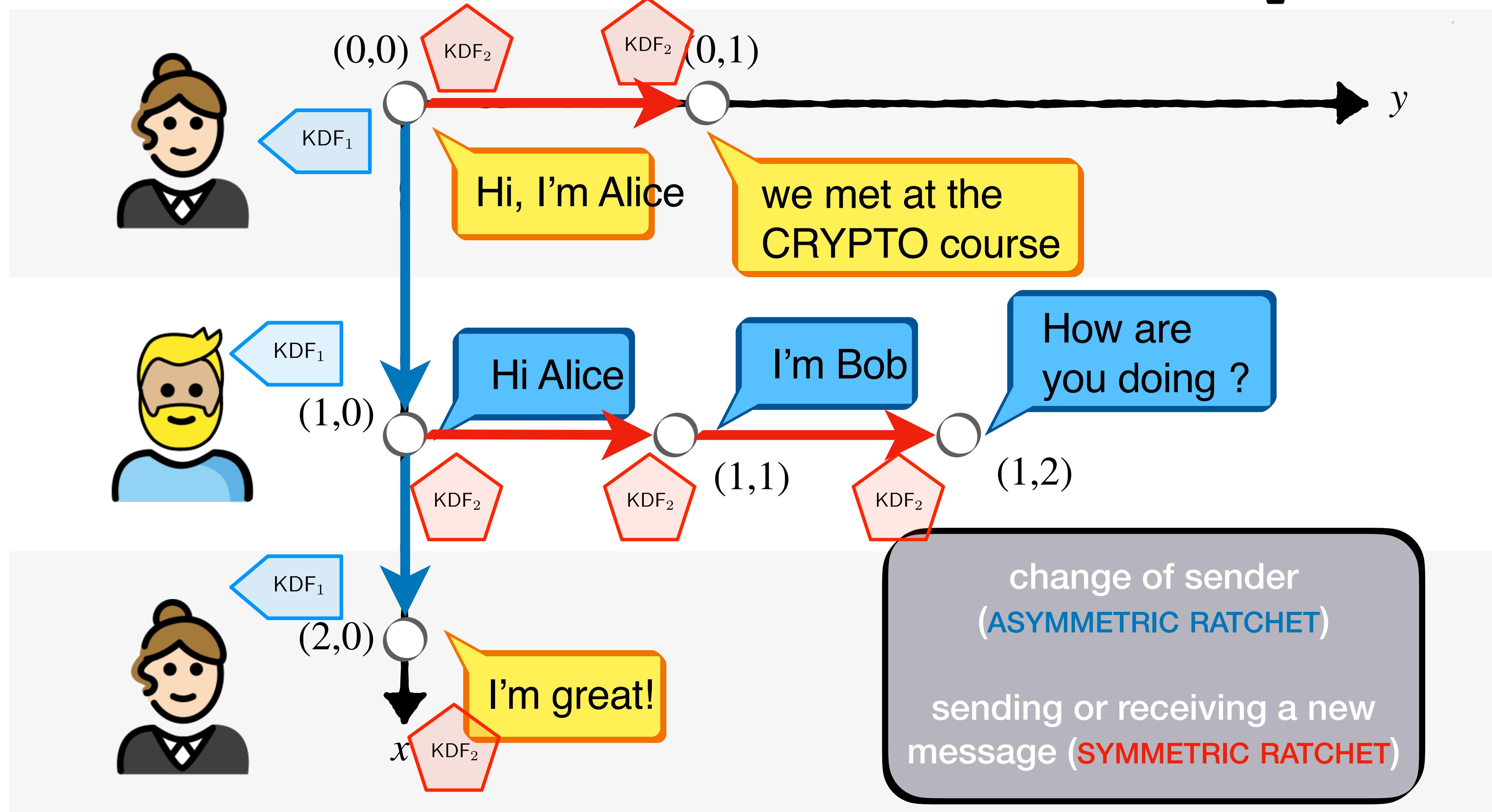
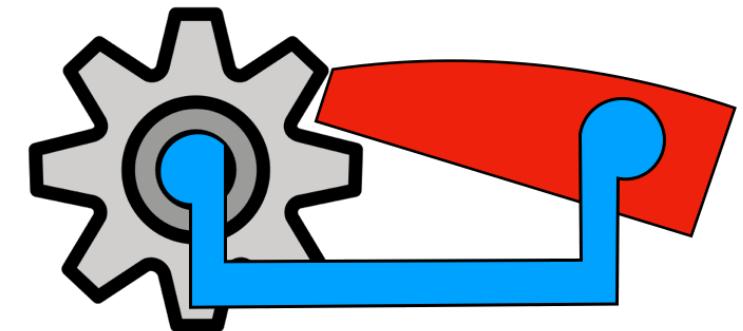
$$rchsk_A^{(0)} = z \leftarrow \$ - \mathbb{Z}_q$$

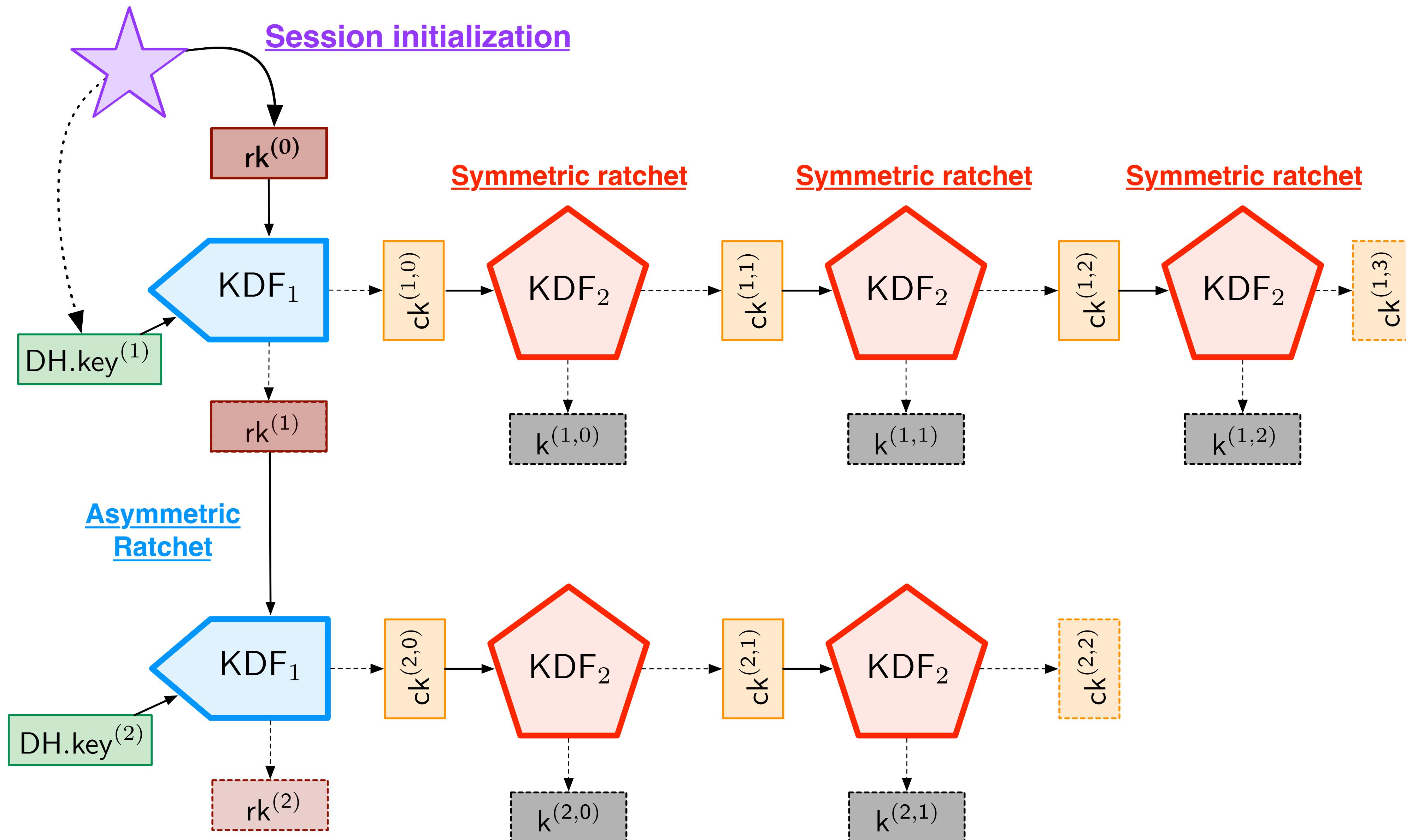
$$rchpk_A^{(0)} = g^z \in \mathbb{G}$$

Signal: Messaging



The Time Line of Asynchronous Messaging





Lecture Agenda

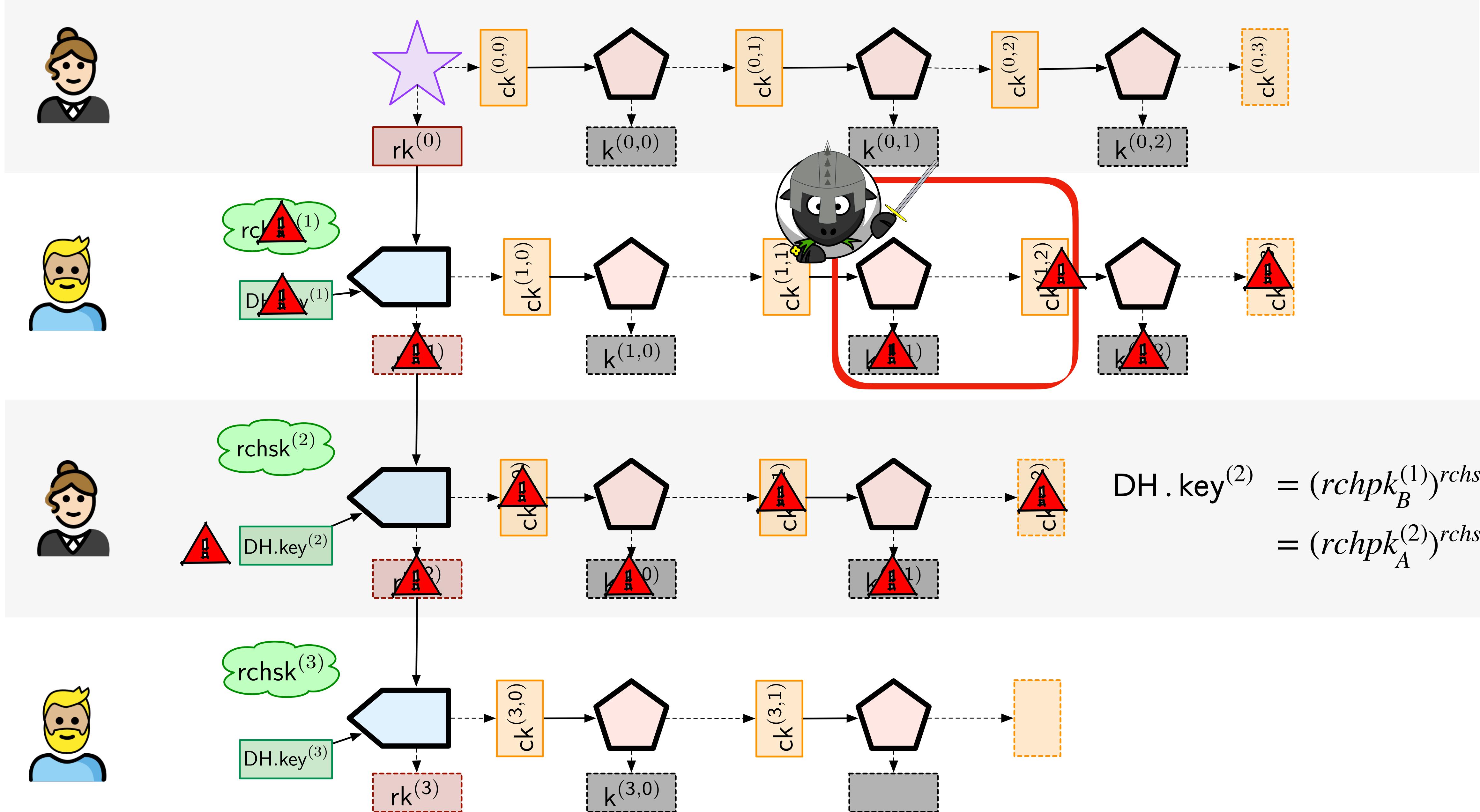
Secure Instant Messaging

- **Security Notions**
 - Backward/Forward Security
- **Tools**
 - AEAD
 - X3DH Protocol
- **The Signal Protocol**
 - Double Ratchet Mechanism
- Session Hijacking Attack

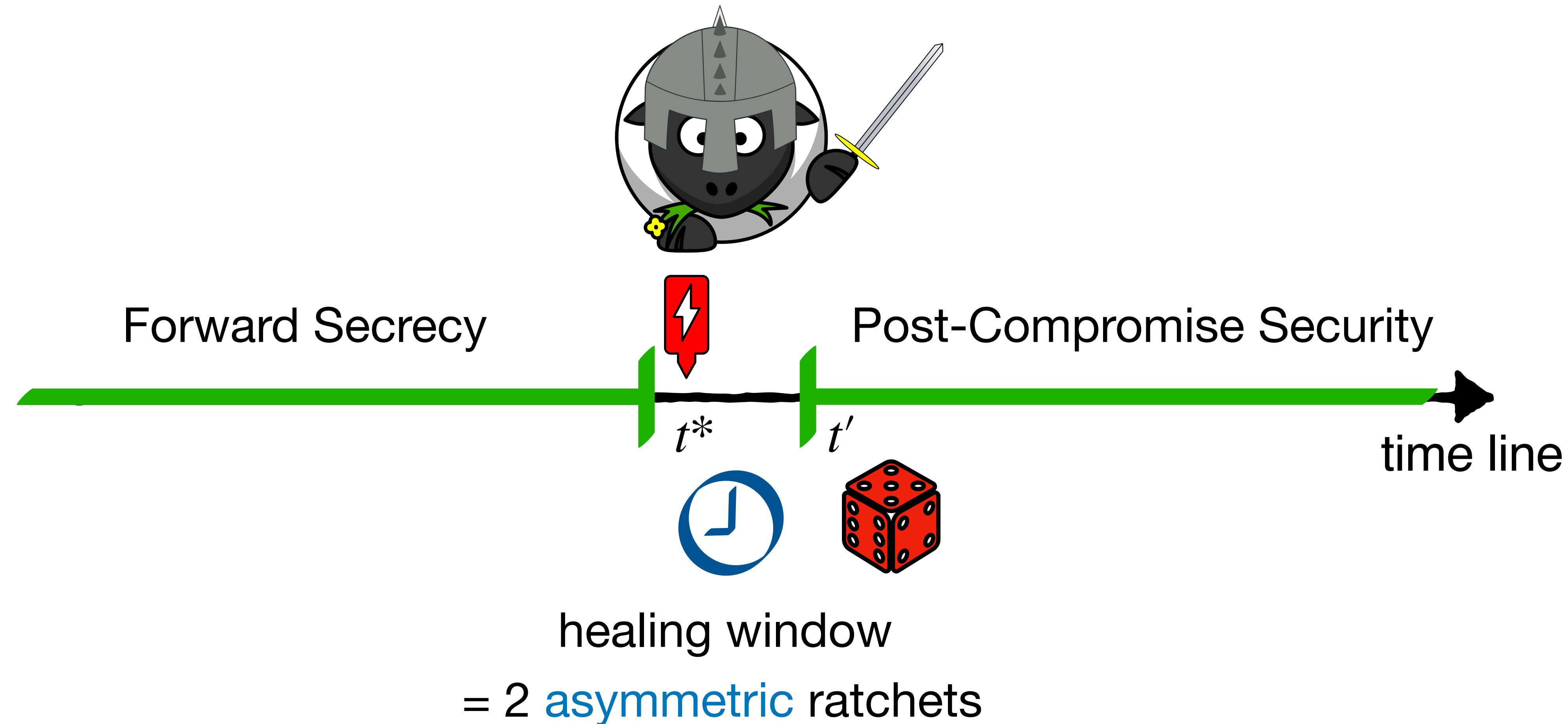
Recap

- Module 2
- Group Theory

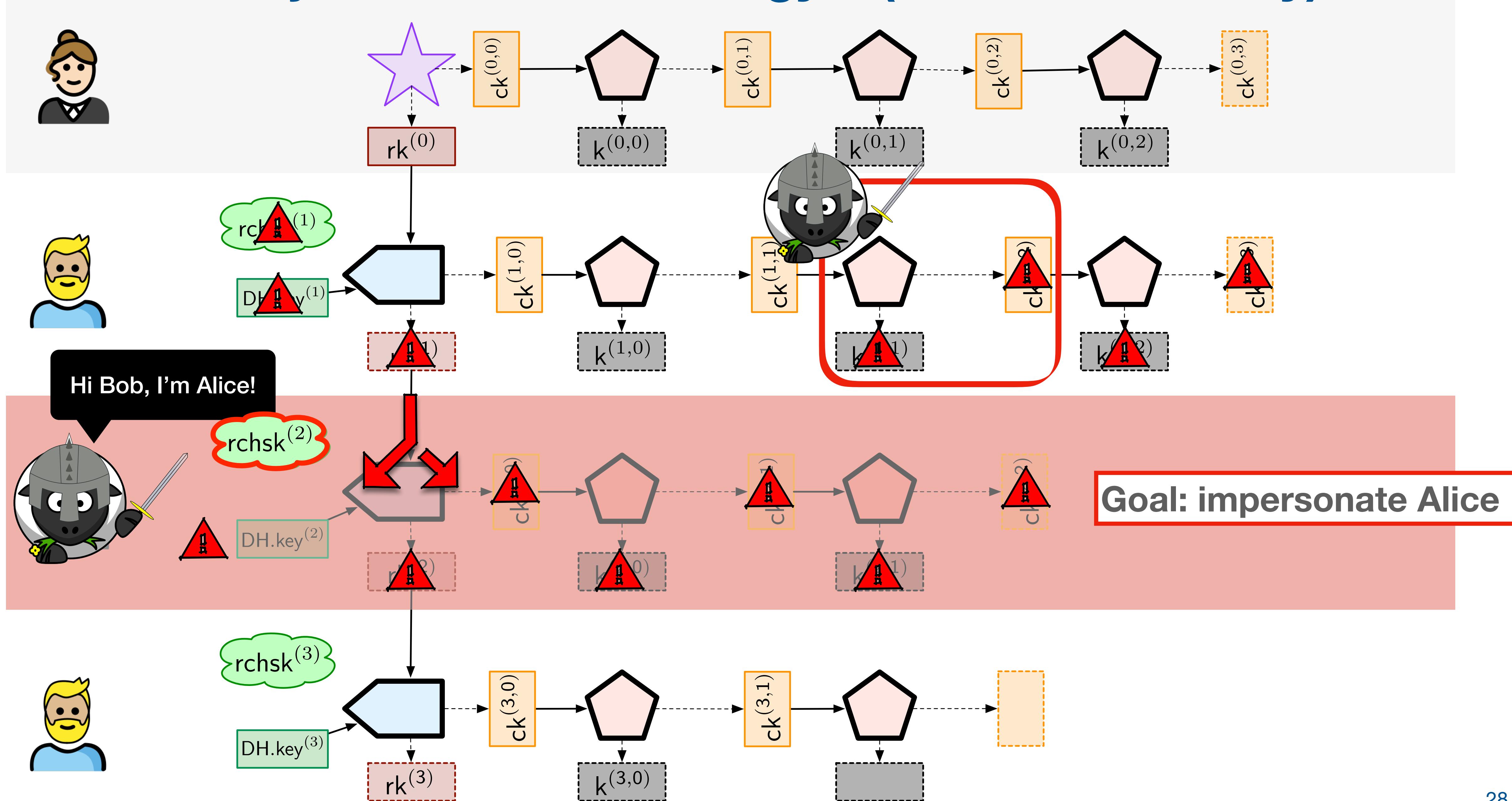
The Impact of Reveal Attacks (Passive Adversary)



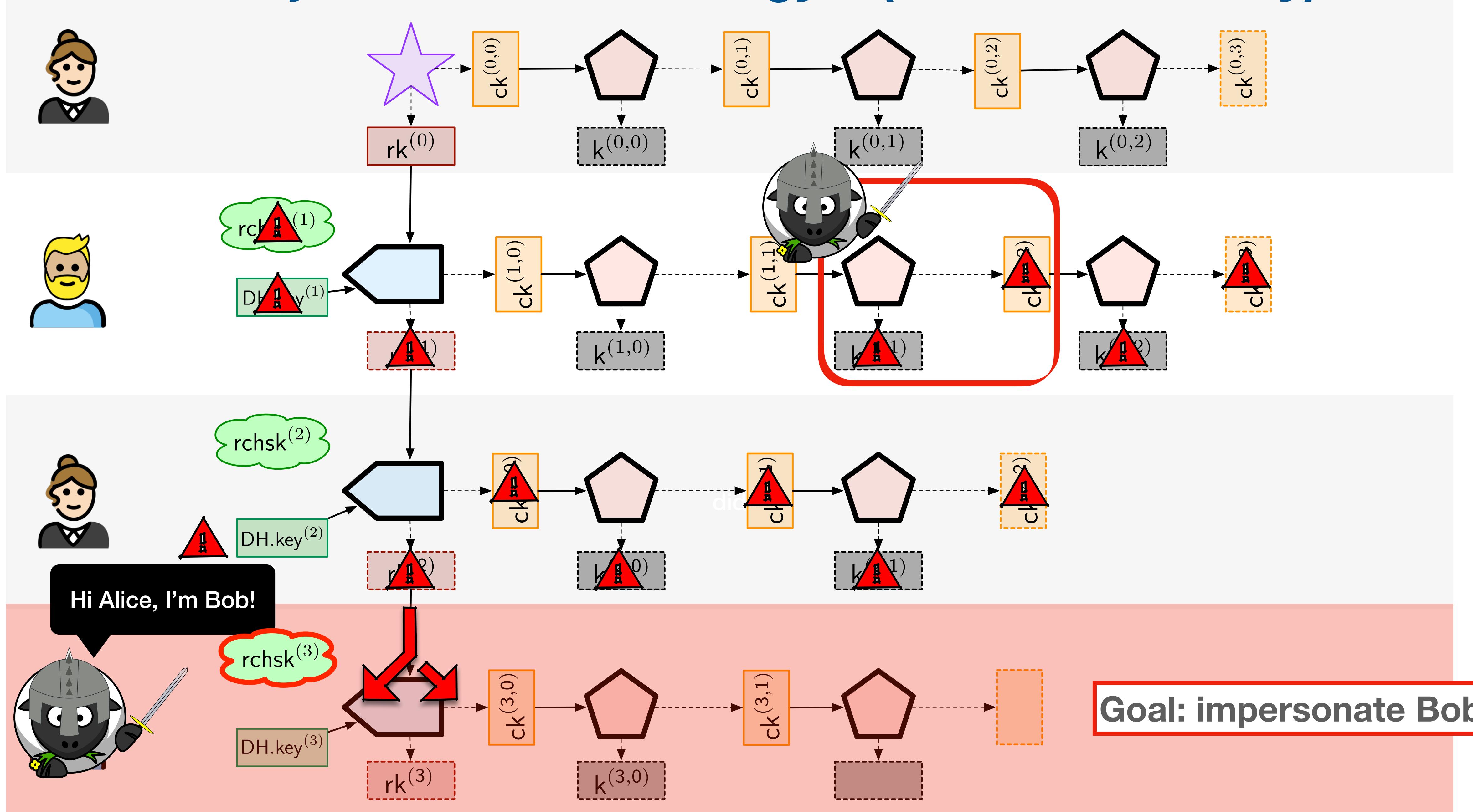
Healing Window for Reveal Attacks



A Reveal & Hijack Attack - Strategy 1 (Active Adversary)



A Reveal & Hijack Attack - Strategy 2 (Active Adversary)



Summary of the Attacks



Gets access to Bob's device at time (x, y)

... and hence the keys: $\text{ck}^{(x,y)}$, rk^{x+1} and $\text{rchsk}_B^{(x)}$ which are present at Bob's device

- Thereby the attacker can derive (**symmetric ratchets**)
 - $\text{ck}^{(x,y+i)}$
 - $\text{k}^{(x,y+i)}$
- Together with the public keys of the protocol (rchpk_A^{x+1}) the attacker can also derive (**asymmetric ratchets**)
 - $\text{rk}^{(x+1)}$
 - $\text{DH}.\text{key}^{(x+1)}$
 - $\text{ck}^{(x,y+i)}$
 - $\text{k}^{(x,y+i)}$
- Passive adversary
 - After 2 **asymmetric** ratchets the adversary can't derive the keys anymore (**healing window**)
- Active adversary
 - Impersonate Alice
(pretend to be Alice to Bob and removing Alice from the conversation)
 - Or impersonate Bob
(pretend to be Bob to Alice and removing Bob from the conversation)

Lecture Agenda

Secure Instant Messaging

- **Security Notions**
 - Backward/Forward Security
- **Tools**
 - AEAD
 - X3DH Protocol
- **The Signal Protocol**
 - Double Ratchet Mechanism
 - Session Hijacking Attack

Recap

- Module 2
- Group Theory

Recap: Module 2

Key Exchange

Diffie Hellman

MiM-attack

Public Key Encryption

Trapdoor One Way functions

IND-CPA

RSA

IND-CCA

EIGamal

Homomorphic

Security Assumption

DL

CDH

DDH

Secure Messaging

Signal Protocol

Backward Security

Forward Security

Ratchets

Digital Signatures

RSA-Sign

EC-DSA

EUF-CMA

Hash-Sign Paradigm

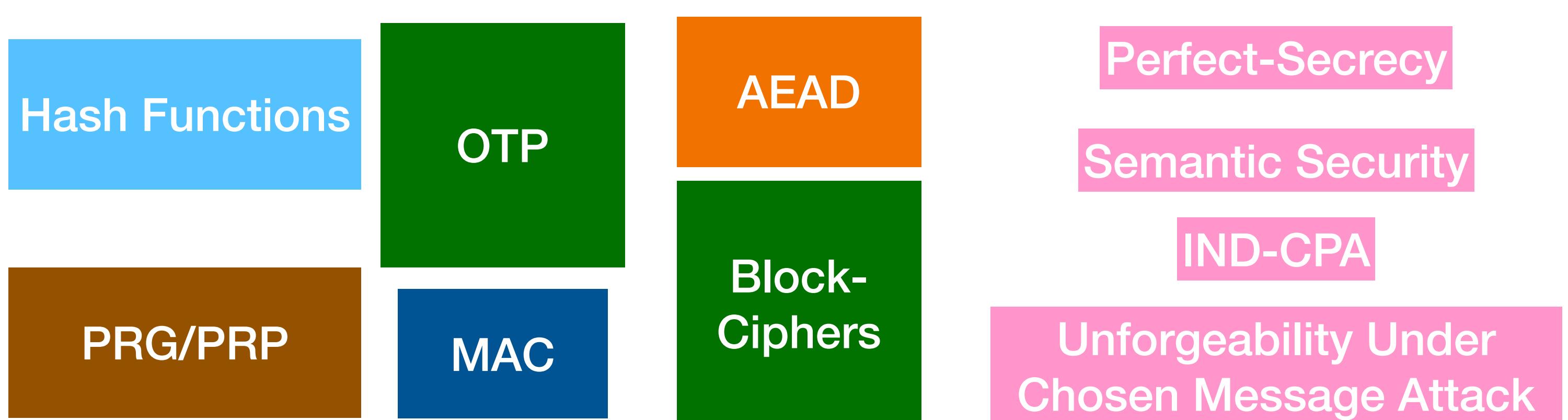
Group Theory

Recap: Group Theory

Group Theory

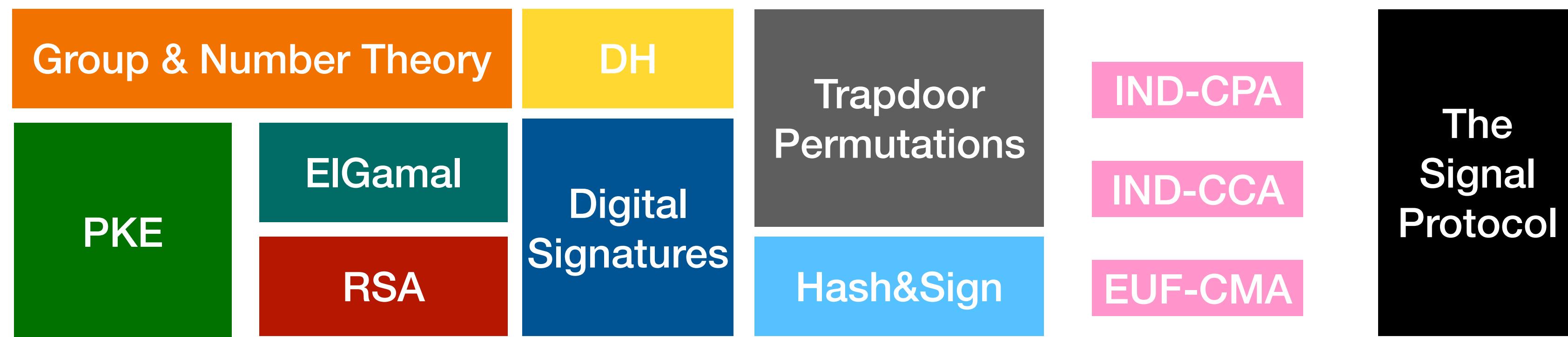
- **What is a group?**
- **What is the difference between \mathbb{Z}_N and \mathbb{Z}_N^***
- **Group generators**
 - Cyclic groups
- **Size of a group:**
 - Euler's Totient function
 - Euler's Theorem
 - Fermat's Little Theorem
- **Subgroups**
 - Order of a subgroup

In Module 1



Now you can understand ~70% of the cryptographic tools used nowadays

In Module 2



Now you can understand +25% of the cryptographic tools used nowadays

Module 3 : Cryptographic Protocols

& Post Quantum Crypto

Cryptographic Protocols

A **cryptographic protocol** performs a security-related feature by combining cryptographic methods (primitives). It often involves two or more parties that interact with one another.



$$= \text{AEAD} + \text{DH} + \text{Digital Signatures} + \text{KDF}$$

Lecture Agenda

Recap & Cryptographic Protocols

Secret Sharing Scheme

- Additive Secret Sharing
- Replicated Secret Sharing
- Facts on Polynomials
- Shamir Secret Sharing Scheme
- Threshold Cryptography

Commitment Schemes

- Hiding & Binding
- Pedersen Commitment - Proof
- Hash-Based Construction

Sharing a Secret...Why?

Dozens of cryptography libraries vulnerable to private key theft

Ben Dickson 28 June 2022 at 15:38 UTC

Updated: 29 June 2022 at 07:34 UTC

Hackers stole over \$4 billion in cryptocurrencies this year — Here's a full list of the biggest crypto heists in 2021

■ MADANA PRATHAP

| DEC 29, 2021, 17:47 IST



How to Best Store a Secret (Key)?

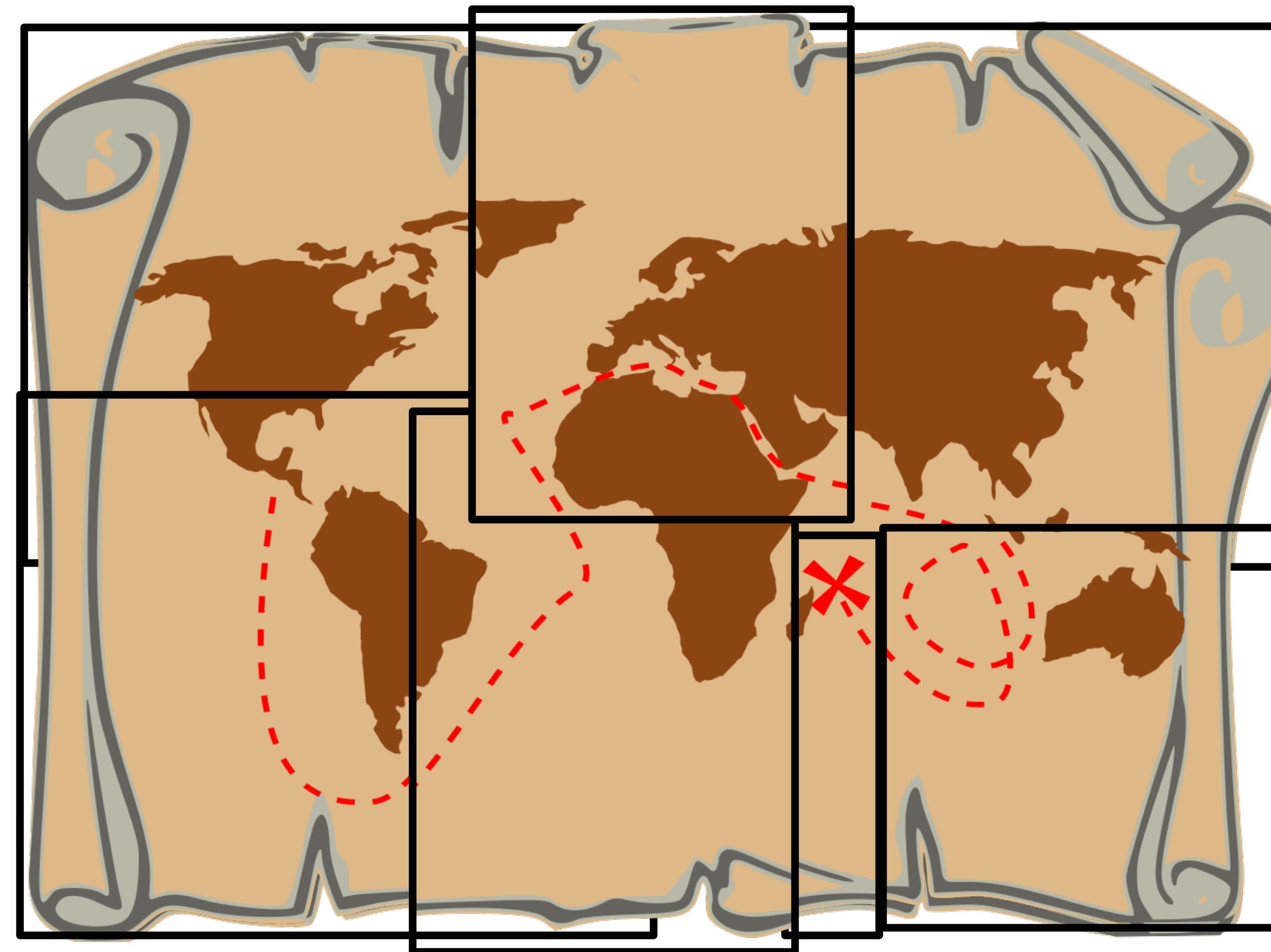
Property Wishlist

1. The secret should be readily **available**
2. The secret should be kept **secure**

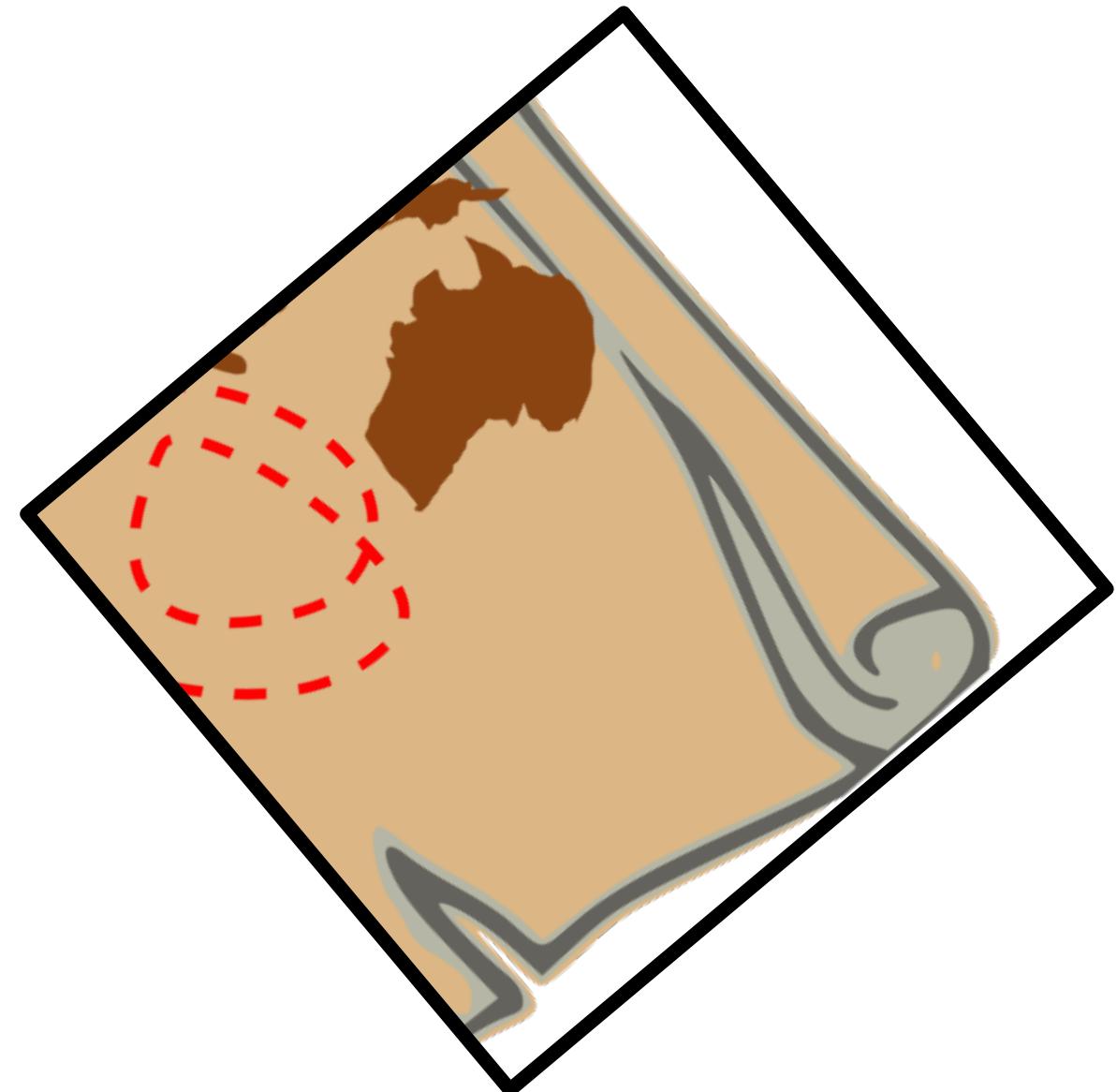
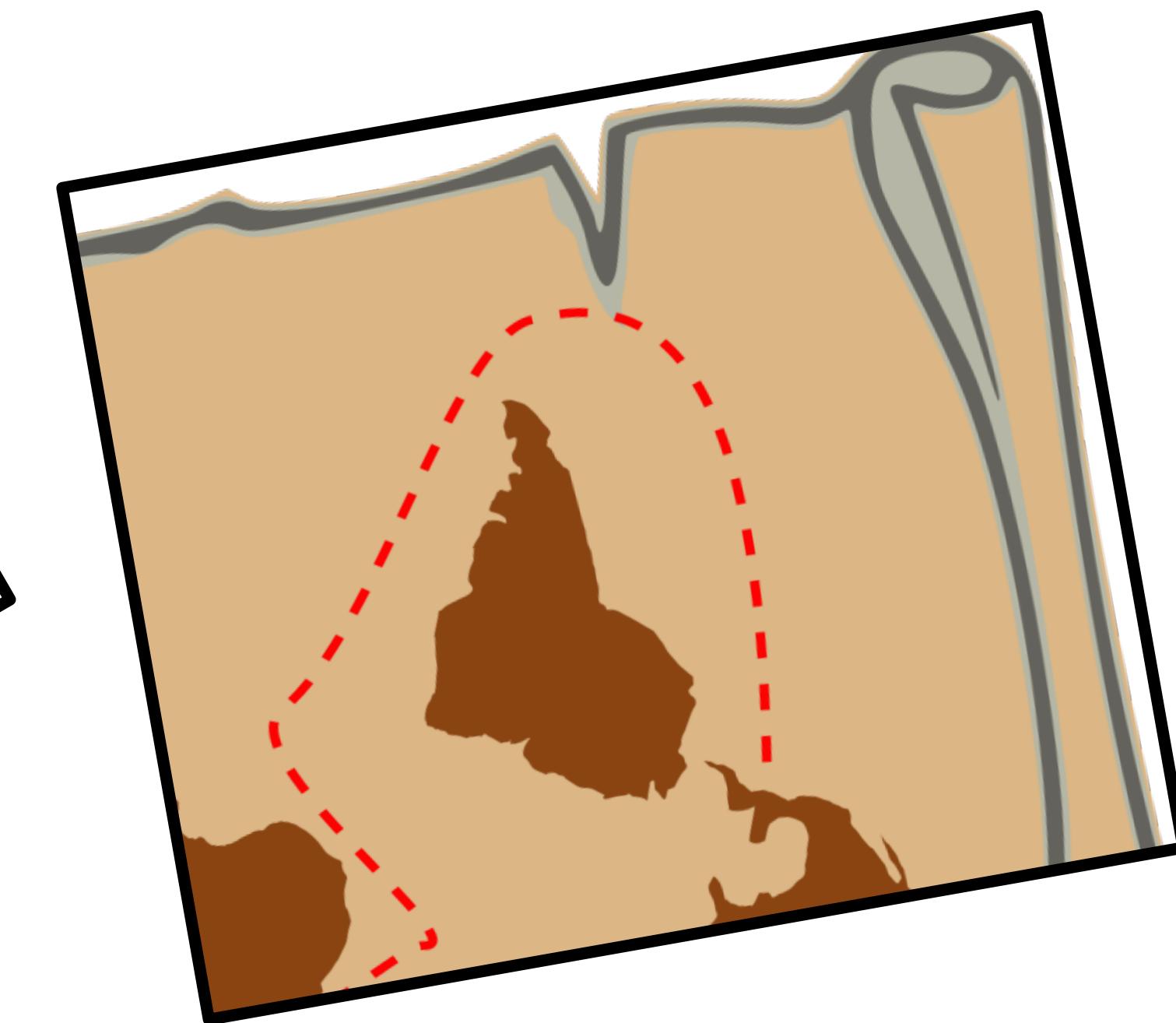
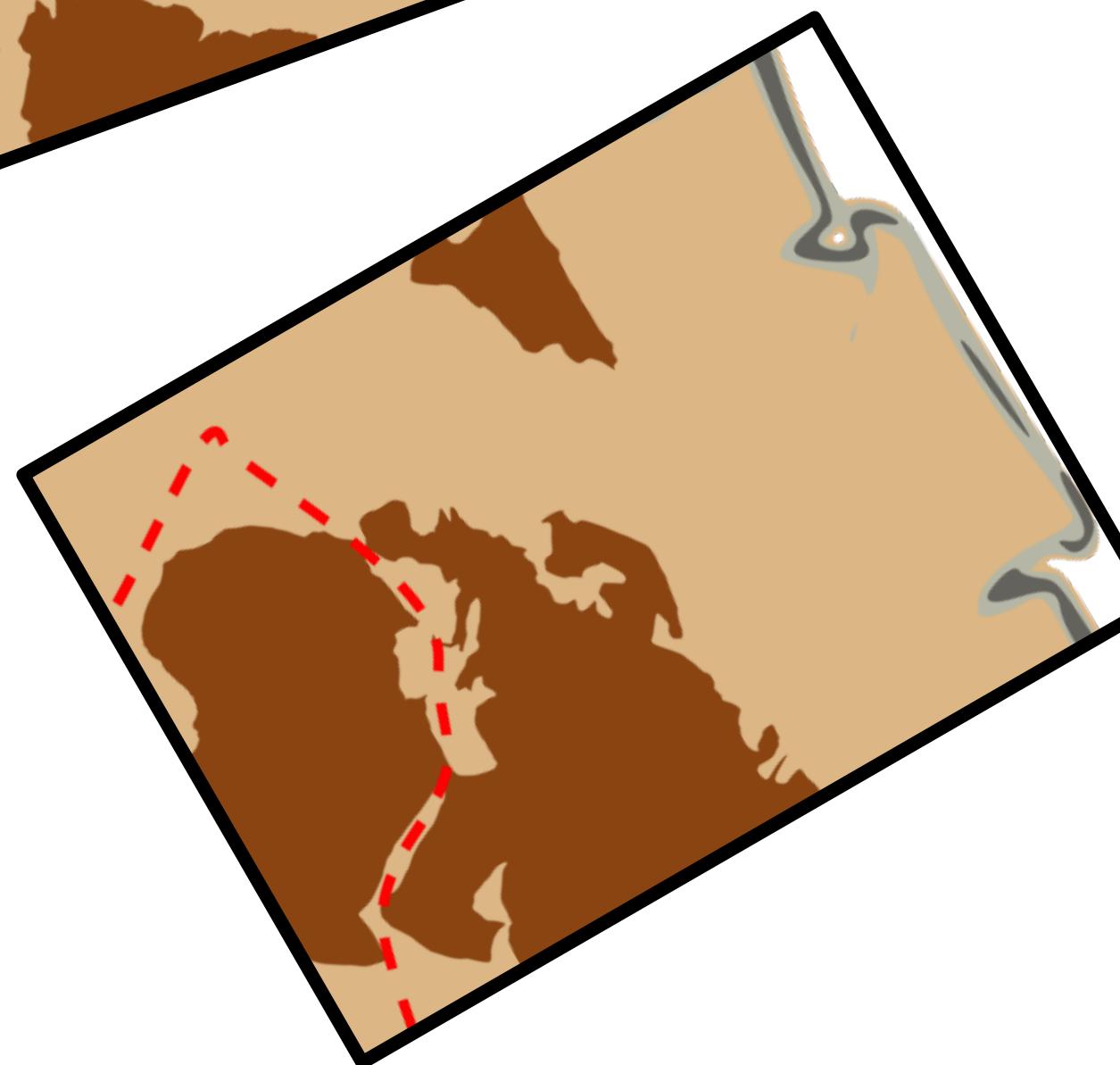
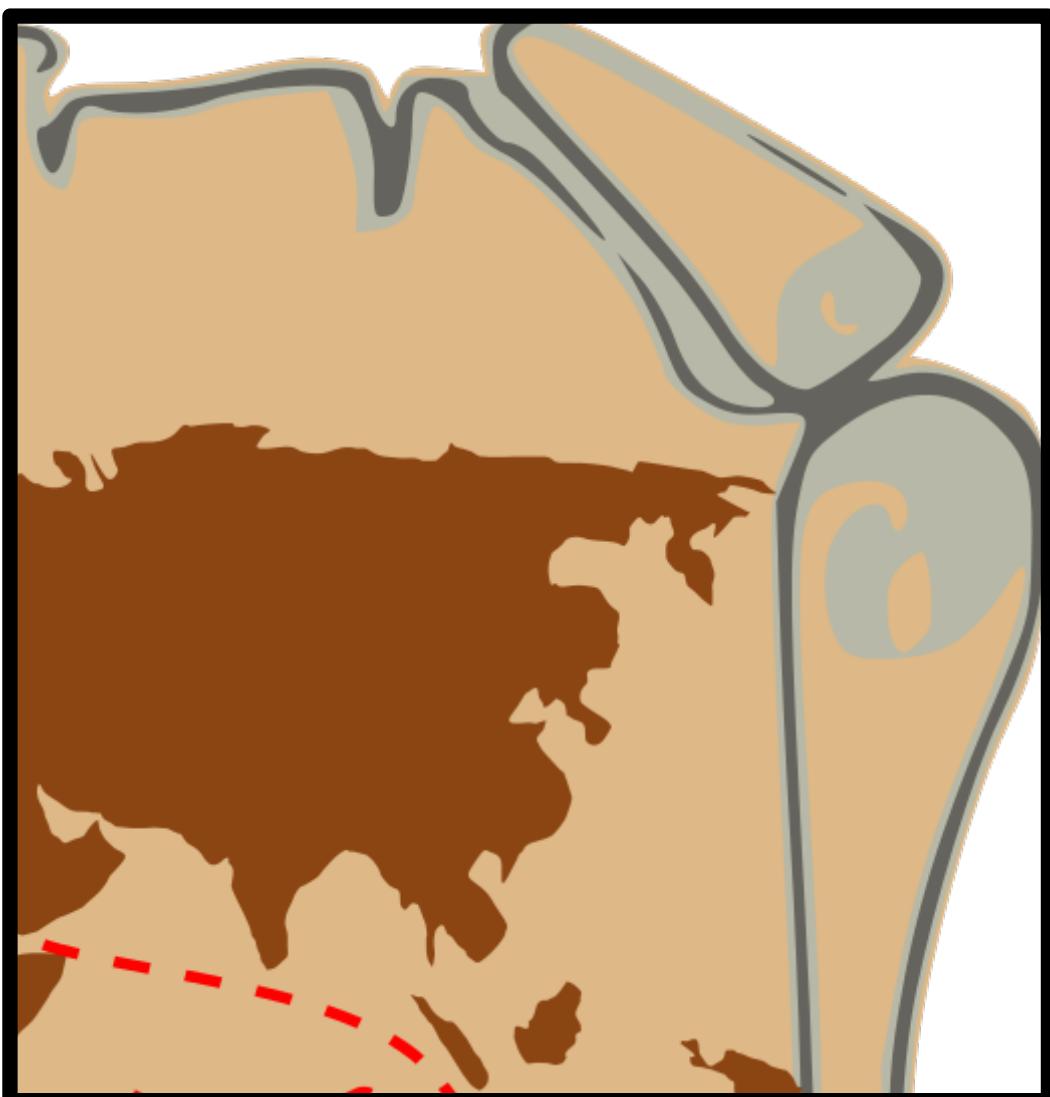
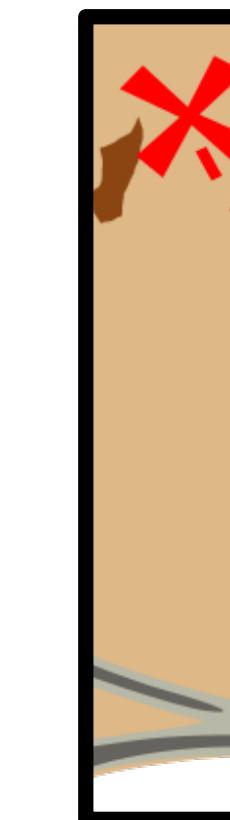
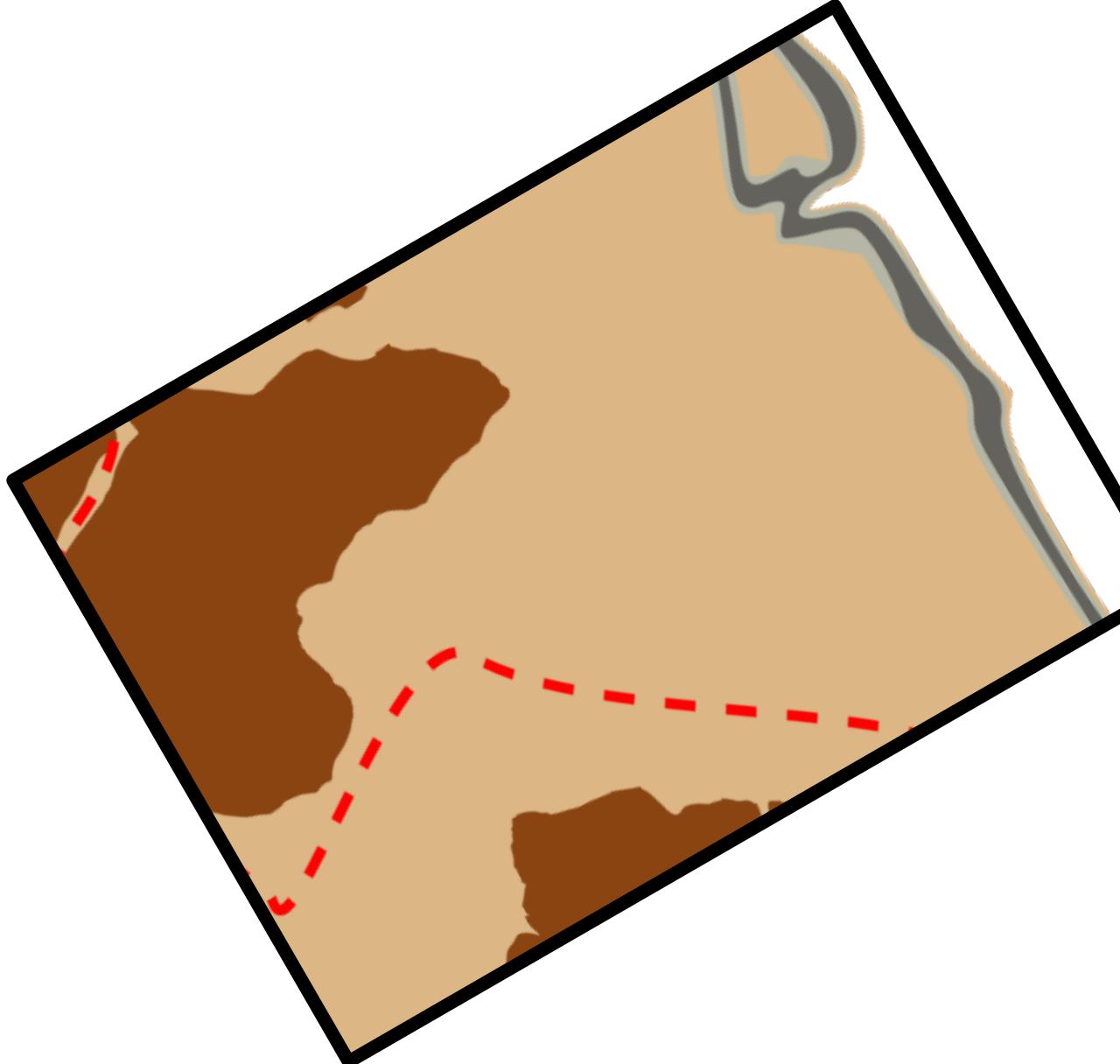
store it in multiple locations

limit the chance of ending up in the wrong hands

Secret Sharing

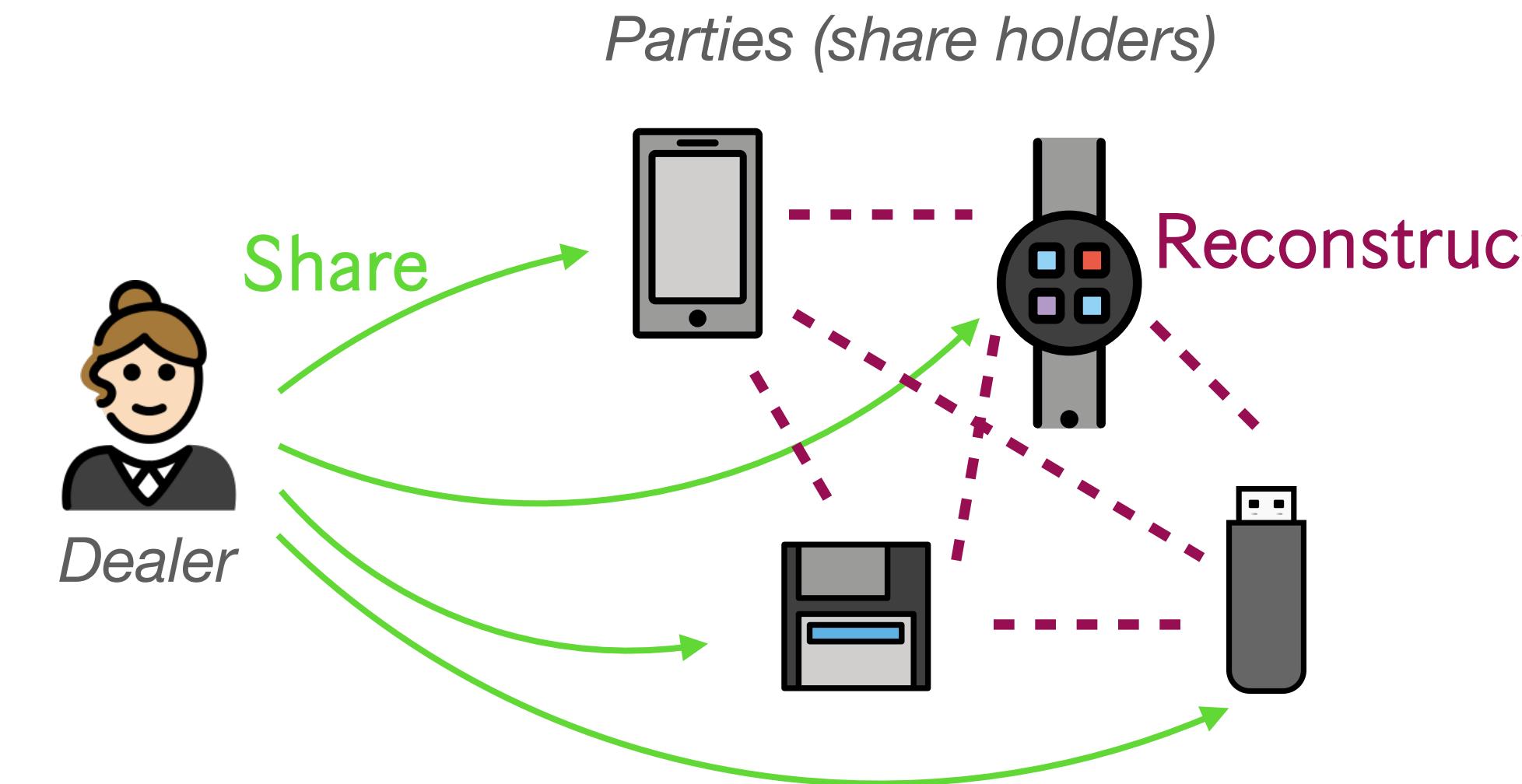


Secret Sharing



What are the implications of “splitting” a cryptographic key in this way?

How To Formalise This Notion?

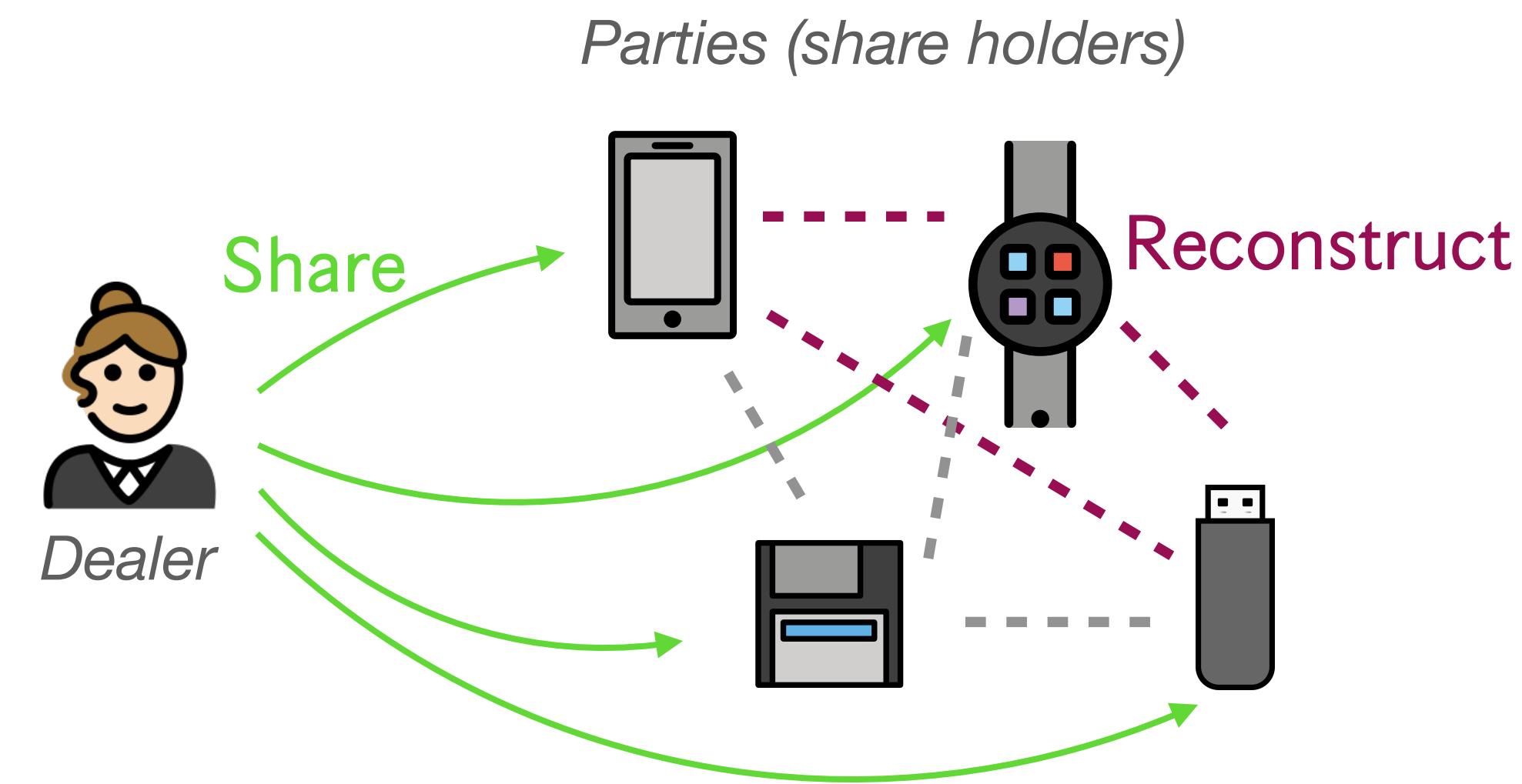


$\text{Share}(s) \rightarrow \{s_1, \dots, s_n\}$ given a secret s , outputs n shares of the secret.

$\text{Reconstruct}(s_1, s_2, \dots, s_n) = s$ deterministic, given all n shares computes the secret s .

🧐 can we do better? Can we make it so that any subset of t share can be used to reconstruct the secret s ?

Defining Threshold Secret Sharing



Definition: Threshold Secret Sharing

A t -out-of- n -secret sharing scheme consists of two efficient algorithms:

- $\text{Share}(s) \rightarrow \{s_1, \dots, s_n\}$ is possibly randomised, it takes as input a secret s , and outputs n shares of the secret.
- $\text{Reconstruct}(s_{i_1}, s_{i_2}, \dots, s_{i_t}) = s$ is deterministic, it takes as input a set of $t \leq n$ distinct shares, and computes the secret s .

Defining Threshold Secret Sharing

t -out-of- n -secret sharing scheme:

- $\text{Share}(s) \rightarrow \{s_1, \dots, s_n\}$
- $\text{Reconstruct}(s_{i_1}, s_{i_2}, \dots, s_{i_t}) = s$

Properties of a Secret Sharing Scheme

1. **t -correctness:** any set of t parties can reconstruct s (together)

$$\Pr[\text{Reconstruct}(s_{i_1}, \dots, s_{i_t}) = s \mid (s_1, \dots, s_n) \leftarrow \text{Share}(s)] = 1 \text{ for all } \{i_1, \dots, i_t\} \subset \{1, \dots, n\}$$

2. **$(t - 1)$ -security:** any subset of *less than t* parties cannot compute s  *how to formalise this?*

For all secrets $s \neq s'$ and for all sets $J \subset \{1, \dots, n\}$ with $|J| < t$ it holds that

$$\{\{s_j\}_{j \in J} \mid (s_1, \dots, s_n) \leftarrow \text{Share}(s)\} \approx \{\{s_j\}_{j \in J} \mid (s_1, \dots, s_n) \leftarrow \text{Share}(s')\}$$

A Simple Construction: (*n* out of *n*) Additive Secret Sharing

Additive *n*-out-of-*n* Secret Sharing: Let N be a large integer (publicly known) and $s \in \mathbb{Z}_N$ be the secret value to be shared.

- Share(s) : $s_1, \dots, s_{n-1} \leftarrow \$\mathbb{Z}_N, s_n = s - (s_1 + \dots + s_{n-1}) \pmod N$, return $s_1, \dots, s_n \in \mathbb{Z}_N$
- Reconstruct(s_1, \dots, s_n): compute $s = s_1 + \dots + s_n \pmod N$

🧐 *is this a t -out-of- n secret sharing scheme?*

- ✓ 0. The algorithms are efficiently computable
- ✓ 1. **t —correctness:** any set of t ($= n$) parties can reconstruct s (together)
- ✓ 2. **$(t - 1)$ -security:** any subset of less than n parties cannot compute s

A More Interesting Construction: 2-out-of-3 Secret Sharing

Replicated Secret Sharing: Fix $n = 3$, $t = 2$, $s \in \mathbb{Z}_N$

- **Share(s)** : pick random $x_1, x_2 \leftarrow \mathbb{Z}_N$, compute $x_3 = s - x_1 - x_2 \pmod{N}$,
create the shares as sets $s_1 = \{(2, x_2), (3, x_3)\}$, $s_2 = \{(1, x_1), (3, x_3)\}$, $s_3 = \{(1, x_1), (2, x_2)\}$,
return $s_1, s_2, s_3 \in (\{1,2,3\} \times \mathbb{Z}_N)^2$
- **Reconstruct(s_i, s_j)**: compute s as the sum of the elements in $s_i \cup s_j$ modulo N
(e.g., $s_1 \cup s_2 = \{(2, x_2), (3, x_3), (1, x_1)\}$, $s = x_1 + x_2 + x_3 \pmod{N}$)

- 0. **efficiency**
- 1. **t -correctness**
- 2. **$(t - 1)$ -security**

- 👍 any set of $t = 2$ parties can reconstruct the secret s
- 👍 one party alone cannot retrieve s
- 👎 (not optimal) to secret share **one** value in \mathbb{Z}_N each party
needs to store **two** values in \mathbb{Z}_N (and additionally, their index)

🧐 *can we do better than this?*

Recall a Few Facts on Polynomials

useful for Tasks 1 & 6 in HA3

A degree d polynomial $f(x) = a_d x^d + a_{d-1} x^{d-1} + \dots + a_0$ with $a_d \neq 0$ can be defined by choosing $d + 1$ values.

Given d points, there are many polynomials of degree d that pass through that set of points

Given $d + 1$ points, there is **only one** polynomial of degree d that passes through those $d + 1$ points



how can we compute it?

Shamir t -out-of- n Secret Sharing (Intuition)

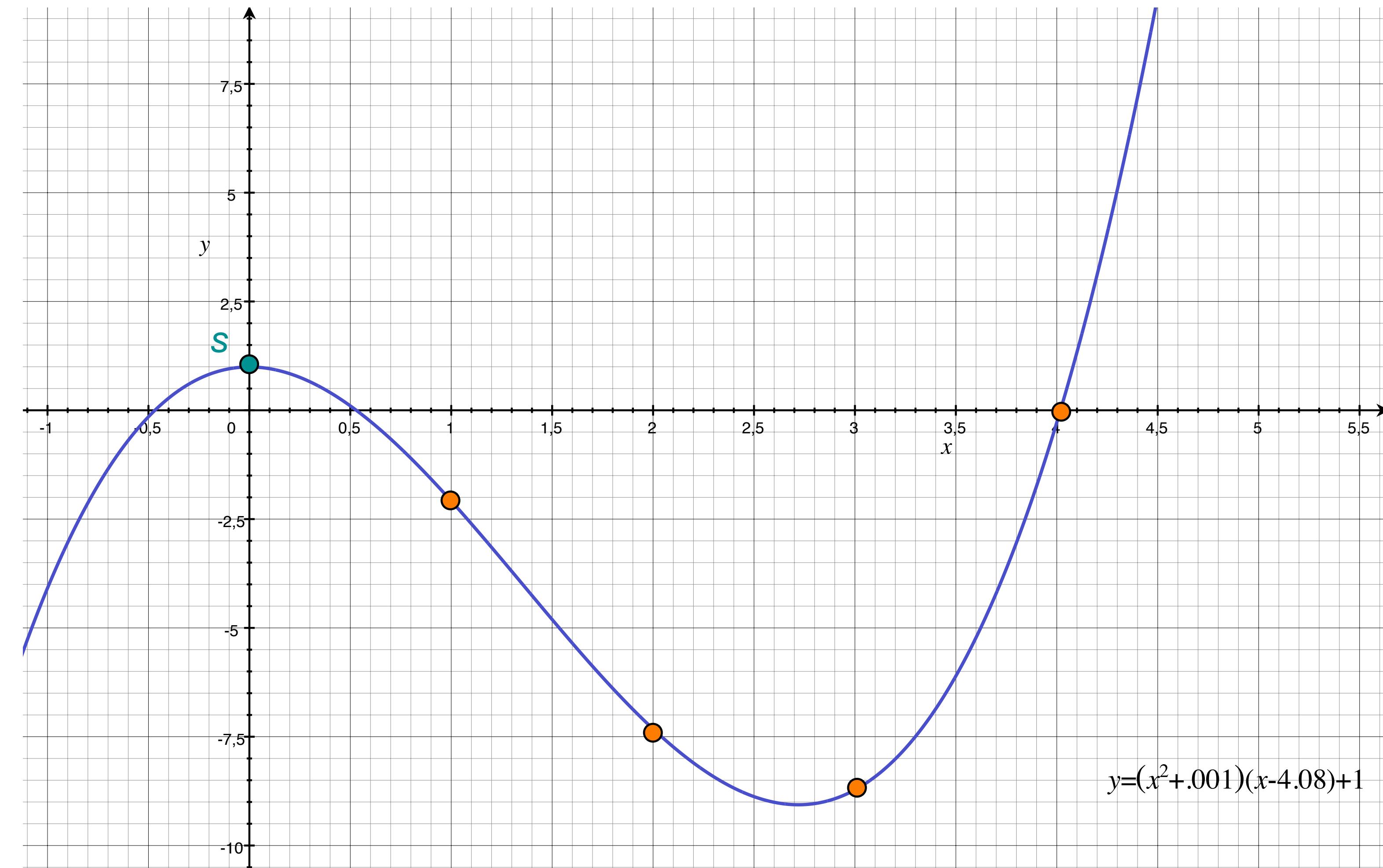
let $s = 1$ be the secret value

construct a random polynomial of degree $t - 1$ passing by the point $(0, s)$

compute the share for party P_i as $s_i = f(i) \bmod p$ for $i \in \{1, 2, 3, \dots, n\}$

👀 the shares s_i look completely random

but if you interpolate $t = 4$ shares, there exists only one polynomial of degree $3 = t - 1$ that passes by all of them.



the polynomial is exactly f , and $f(0) = s$ is exactly the secret (reconstructed by interpolation from the shares)

Shamir t -out-of- n Secret Sharing Scheme

Share(s) Given the value $s \in \mathbb{Z}_p$, where p is a prime and $p > n$, do:

- sample $t - 1$ random values $a_1, \dots, a_{t-1} \leftarrow \\mathbb{Z}_p with $a_{t-1} \neq 0$
- construct the polynomial $f(x) = s + a_1x + a_2x^2 + \dots + a_{t-1}x^{t-1} \in \mathbb{Z}_p[x]$
- compute the n shares by evaluating $f(x)$ on n distinct points: $s_i = f(i)$ for $i \in \{1, \dots, n\}$
- send the share s_i to party P_i (via a secure channel)

Reconstruct($s_{i_1}, s_{i_2}, \dots, s_{i_t}$) Let $I = \{i_1, \dots, i_t\}$ be the set distinct share indexes do:

- compute the Lagrange coefficients for the set I : $\ell_i^I(0) = \prod_{j \in I \setminus \{i\}} \frac{j}{j - i} \pmod{p}$
- Recover the secret $s = \sum_{i \in I} s_i \cdot \ell_i^I(0) \pmod{p}$

Why Does This (Magic) Work?

- compute the Lagrange coefficients for the set I : $\ell_i^I(x) = \prod_{j \in I \setminus \{i\}} \frac{j - x}{j - i} \pmod{p}$

polynomial identity: $f(x)$ and $\sum_{i \in I} f(i) \ell_i^I(x)$ coincide on $d + 1$ points

[Blackboard]

- the polynomials $\ell_i^I(x)$ only depend on $i, j \in I$ and not on $f(x)$
- the polynomials $\ell_i^I(x)$ exist for every i and every set of $I \subseteq \mathbb{Z}_p$ of cardinality t
- setting $x = 0$ we get $s = f(0) = f(i_1) \ell_{i_1}^I(0) + f(i_2) \ell_{i_2}^I(0) + \dots + f(i_t) \ell_{i_t}^I(0) \pmod{p}$

Is Shamir's Scheme a Secure Secret Sharing?



0. The algorithms are efficiently computable

1. **t -correctness:**

$$\Pr[\text{Reconstruct}(s_{i_1}, \dots, s_{i_t}) = s \mid (s_1, \dots, s_n) \leftarrow \text{Share}(s)] = 1 \text{ for all } i_1, \dots, i_t \subset \{1, \dots, n\}$$

2. **$(t - 1)$ -security:** for any $J \subset \{1, 2, \dots, n\}$, $|J| \leq t - 1$

$$\{\{s_j\}_{j \in J} \mid (s_1, \dots, s_n) \leftarrow \text{Share}(s)\} \approx \{\{s_j\}_{j \in J} \mid (s_1, \dots, s_n) \leftarrow \text{Share}(s')\}$$

Yes! And all properties hold unconditionally (information theoretic security)

But for the scheme to work, we rely on the fact that the n parties have peer-to-peer, confidential and possibly authenticated channels (that can be built with techniques from Module 1 and 2).

And Now What?



Threshold Cryptography

Threshold El-Gamal's Cryptosystem

- $\text{KeyGen}(1^n)$: Sample $\text{sk} \leftarrow \$\{2, \dots, q - 1\}$. Set $\text{pk} = g^{\text{sk}}$ (where g is a generator of a group \mathbb{G} of large prime order q). Use Shamir secret sharing to compute $(\text{sk}_1, \dots, \text{sk}_n) \leftarrow \text{Share}(\text{sk})$. Output pk (public), and send sk_i to party i
- $\text{Enc}(\text{pk}, m)$: Sample $r \leftarrow \$\{2, \dots, q - 1\}$. Output the ciphertext $c = (g^r, m \cdot \text{pk}^r)$ [precisely as in ElGamal]
- $\text{Partial}.\text{Dec}(\text{sk}_i, c)$: Parse $c = (c_1, c_2)$. Compute $d_i = c_1^{\text{sk}_i}$
- $\text{Collaborate}.\text{Dec}(I, \{d_i\}_{i \in I}, c_2)$: [$I \subset \{1, 2, \dots, n\}$ is the set of indexes of collaborating parties]
- Compute $D = \prod_{i \in I} d_i^{\ell_i^I(0)} = g^{r \sum_{i \in I} \text{sk}_i \cdot \ell_i^I(0) \bmod q} = g^{r \cdot \text{sk}}$
this is running Reconstruct in the exponent
Output $m = c_2 \cdot D^{-1}$.

Lecture Agenda

Recap & Cryptographic Protocols

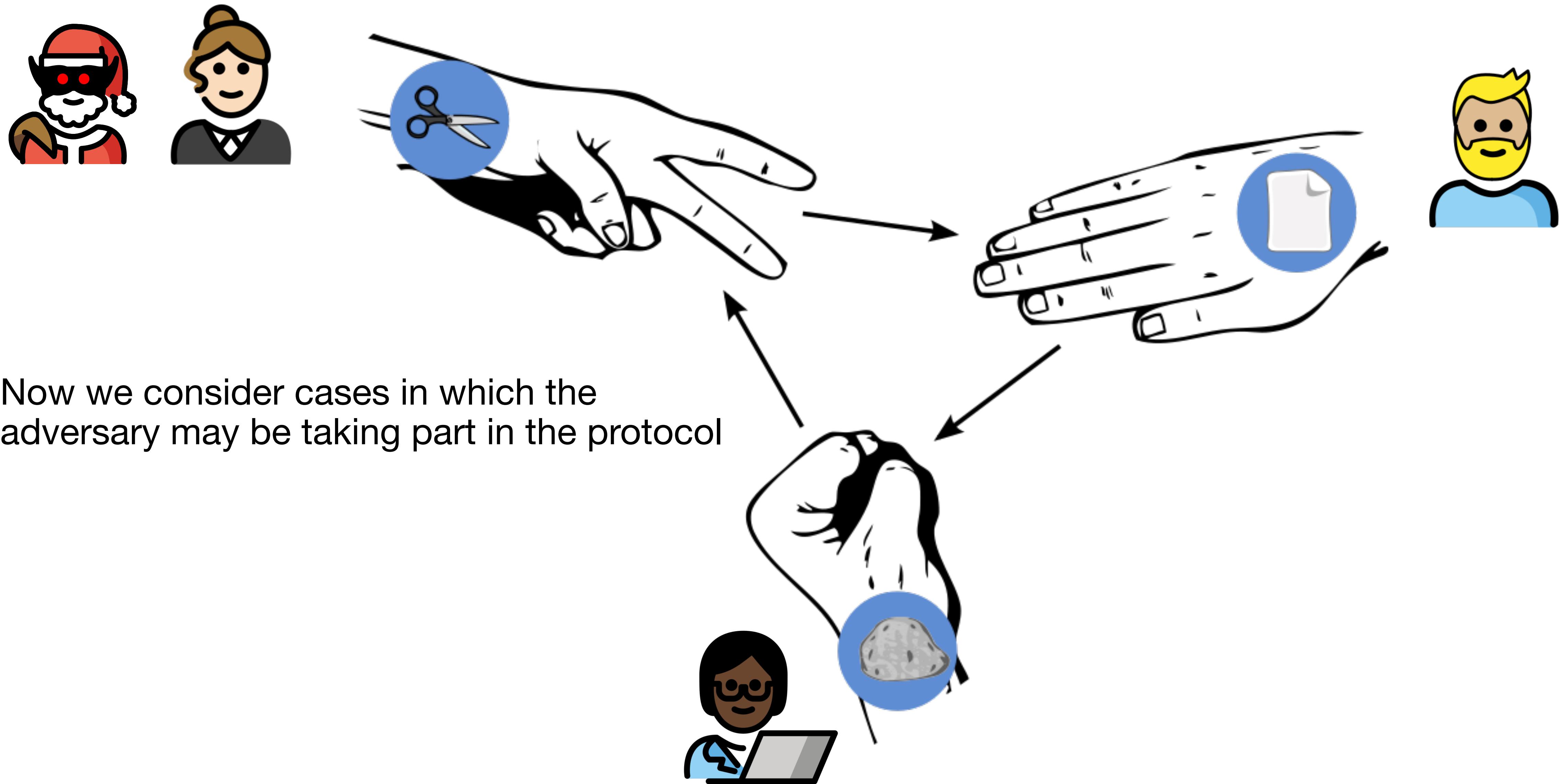
Secret Sharing Scheme

- Additive Secret Sharing
- Replicated Secret Sharing
- Facts on Polynomials
- Shamir Secret Sharing Scheme
- Threshold Cryptography

Commitment Schemes

- Hiding & Binding
- Pedersen Commitment - Proof
- Hash-Based Construction

Use Case: Playing Rock-Paper-Scissors



Rock-Paper-Scissors Over the Internet



🤔 How do we **formalise** the security requirements?

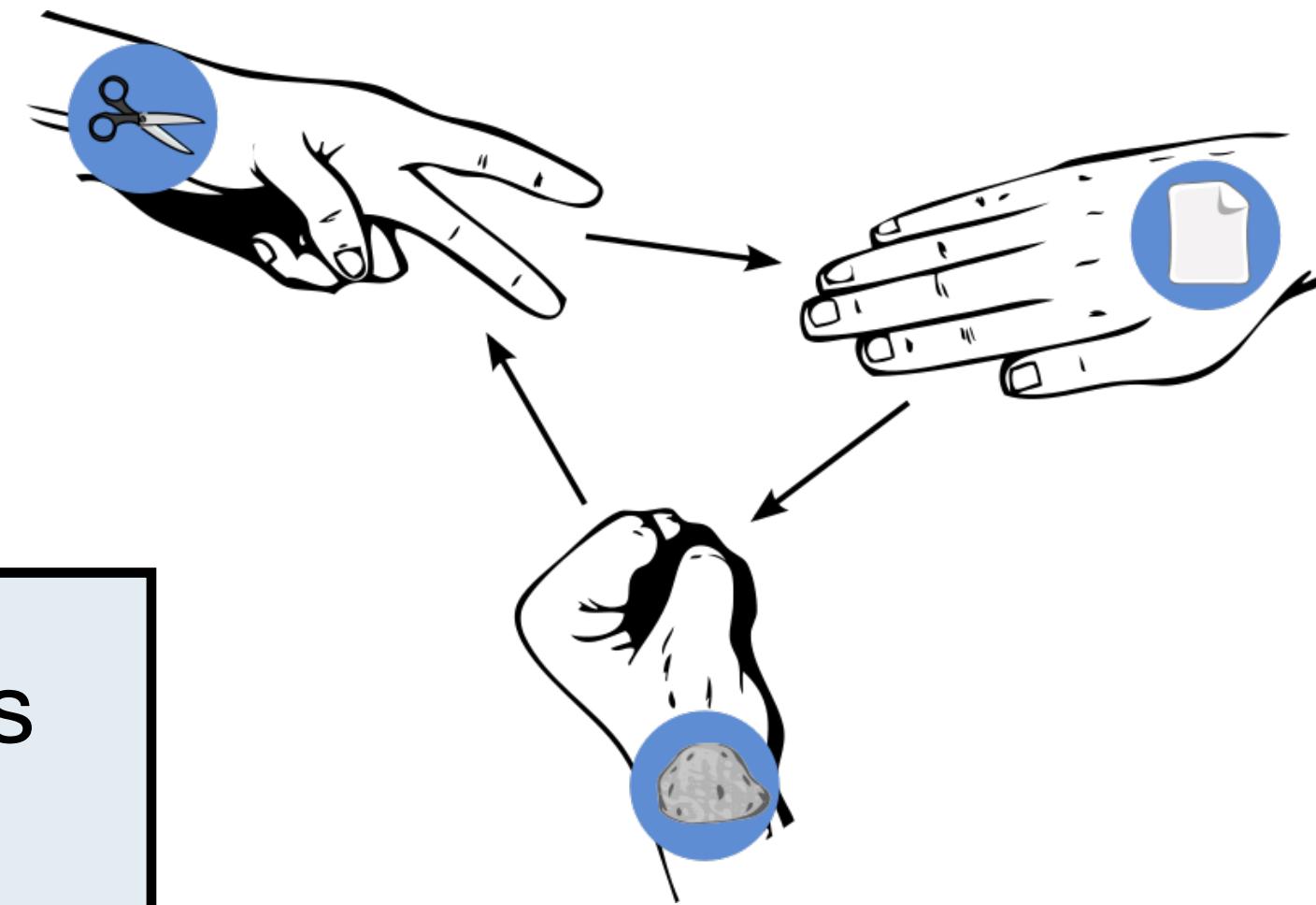
Commitment Schemes Definitions

A **commitment scheme** is an interactive protocol between two parties (sender S and receiver R) and is composed of two phases:

Commit Phase. The sender S commits to a value m by computing $c = \text{Commit}(m, r)$ for a *random* value r , and sends c to the receiver R.

Reveal Phase. The sender S ‘opens’ its commitment by sending (m, r) to the receiver. The receiver R *verifies* that the value c it got during the commit phase matches $\text{Commit}(m, r)$ for the values (m, r) it got during the reveal phase.

... and satisfies the **binding** and **hiding** properties (given next)



Commitment Schemes Definitions - Binding

Intuition: the binding property protects the receiver (S cannot change its mind, once it commits)

A commitment scheme is said to be **binding** if no adversary \mathcal{A} can find two distinct messages $m \neq m^*$ (and randomness r, r^*) that yield the same commitment value: $c = \text{Commit}(m, r) = \text{Commit}(m^*, r^*)$.

$$\Pr[\text{Commit}(m, r) = \text{Commit}(m^*, r^*) \mid m \neq m^* \wedge (m, r, m^*, r^*) \leftarrow \mathcal{A}] \leq \text{negl}(n)$$

n is the bit-length of the randomness

computational
(complexity-based)

vs

information-theoretic
(unconditional)

= 0

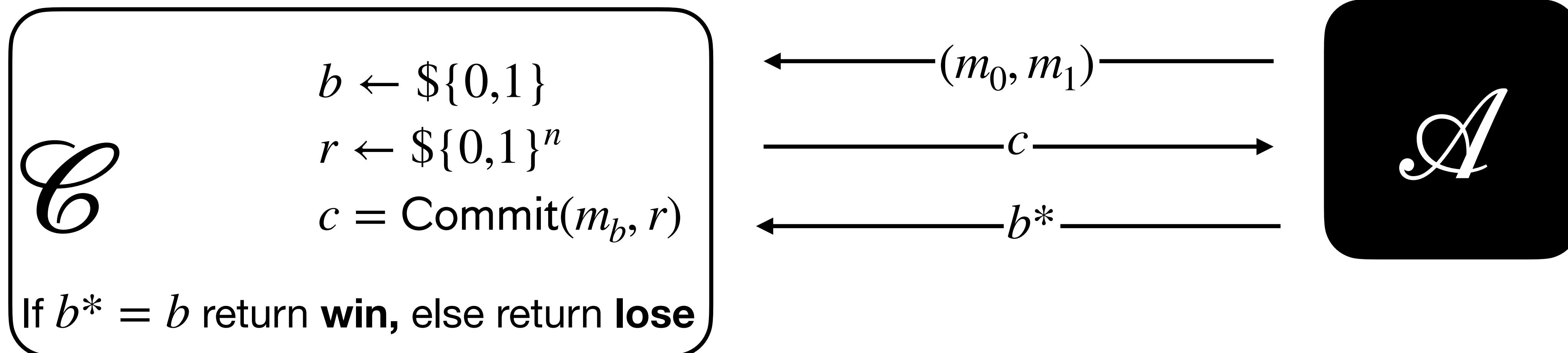
A commitment scheme is called **computationally binding** if \mathcal{A} is a PPT algorithm.

If no such restriction is made the scheme is called **information-theoretically binding**.

Commitment Schemes Definitions - Hiding

Intuition: the hiding property protects the senders (R cannot get the committed m from c)

A commitment scheme is said to be **hiding** if no adversary \mathcal{A} can win the following game better than random guessing.



🧐 have we seen this game structure before?

Commitment Schemes Definitions - Hiding

A commitment scheme is said to be **hiding** if no adversary \mathcal{A} can win the following game with probability noticeably higher than $1/2$:

1. \mathcal{A} outputs two messages m_0 and m_1 .
2. \mathcal{C} selects a random bit $b \leftarrow \{0,1\}$; picks a uniform random randomness value $r \leftarrow \{0,1\}^n$; computes $c = \text{Commit}(m_b, r)$; and returns c to \mathcal{A} .
3. \mathcal{A} outputs a bit b^* as a guess for b .

\mathcal{A} wins the game if $b^* = b$, else \mathcal{A} loses the game.

A commitment scheme is called **computationally hiding** if \mathcal{A} is a PPT algorithm and $\Pr[b = b^*] \leq \frac{1}{2} + \text{negl}(n)$.

If no such restriction is made the scheme is called **information-theoretically hiding** and $\Pr[b = b^*] = \frac{1}{2} + 2^{-n}$

Pedersen Commitment Scheme

Tasks 3,4 in HA3

Let $\mathbb{G} = \langle g \rangle$ be a cyclic multiplicative subgroup of prime order q inside the multiplicative group \mathbb{Z}_p^* .

Let h be a random element in $\mathbb{G} \setminus g$. Let p, q, g, h be all public information.

The **Pedersen** commitment function is defined as:

$$\text{Commit}(m, r) = g^m h^r \pmod{p} = c \quad (\text{for } r, m \in \{0, 1, 2, \dots, q-1\})$$

🧐 **Binding?**

yes, computationally (reduces to DL)

$$\Pr \left[\text{Commit}(m, r) = \text{Commit}(m^*, r^*) \middle| \begin{array}{l} (m, r, m^*, r^*) \leftarrow \mathcal{A}(p, q, g, h) \\ \wedge m \neq m^* \end{array} \right] \leq \text{negl}(\log_2(q))$$

Proof by reduction: \mathcal{A} provides (m, r, m^*, r^*) that yield the same commitment value.

The reduction \mathcal{A}' solves the DL instance $(g, h = g^x)$ by sending $x^* = (m^* - m)(r - r^*)^{-1} \pmod{q}$.

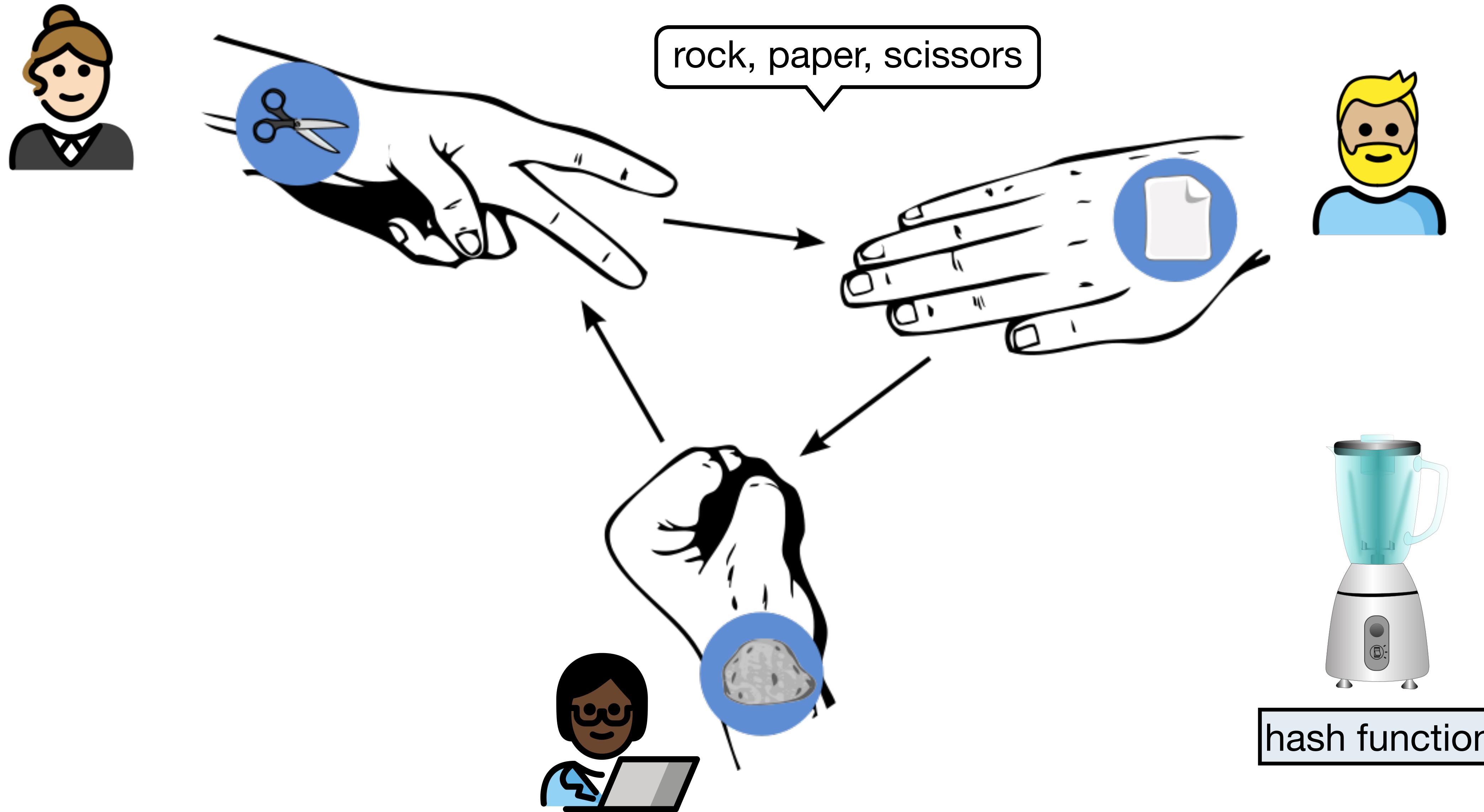
🧐 **Hiding?**

yes, information-theoretically (even if the adversary could solve DL)

$$\Pr \left[b = b^* \middle| \begin{array}{l} (m_0, m_1) \leftarrow \mathcal{A}(p, q, g, h) \\ b \leftarrow \$\{0, 1\}, r \leftarrow \$\mathbb{Z}_q \\ \text{Commit}(m_b, r) = c \\ b^* \leftarrow \mathcal{A}(c) \end{array} \right] = \frac{1}{2} + \frac{1}{q}$$

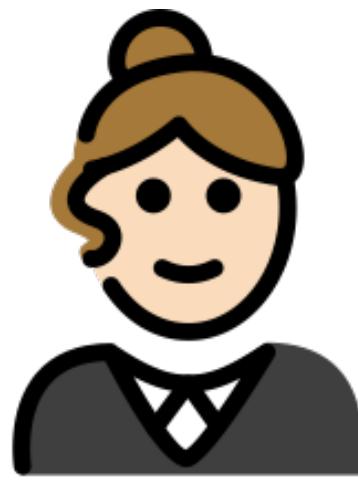
Proof: show that for any commitment c there exist r_0, r_1 s.t. $c = g^{m_0} h^{r_0} = g^{m_1} h^{r_1}$ (similar to OTP).

Let's Construct a Secure Commitment Scheme Using A Cryptographic Hash Function



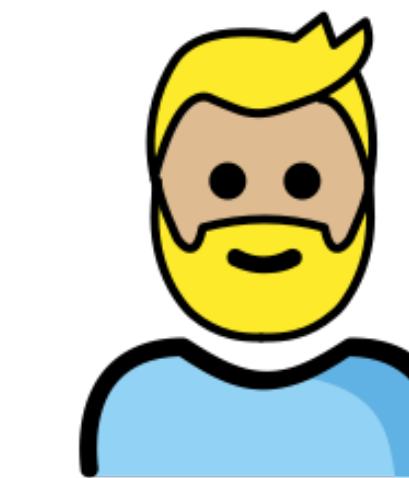
Commitment Schemes: a Simple Construction

hash function

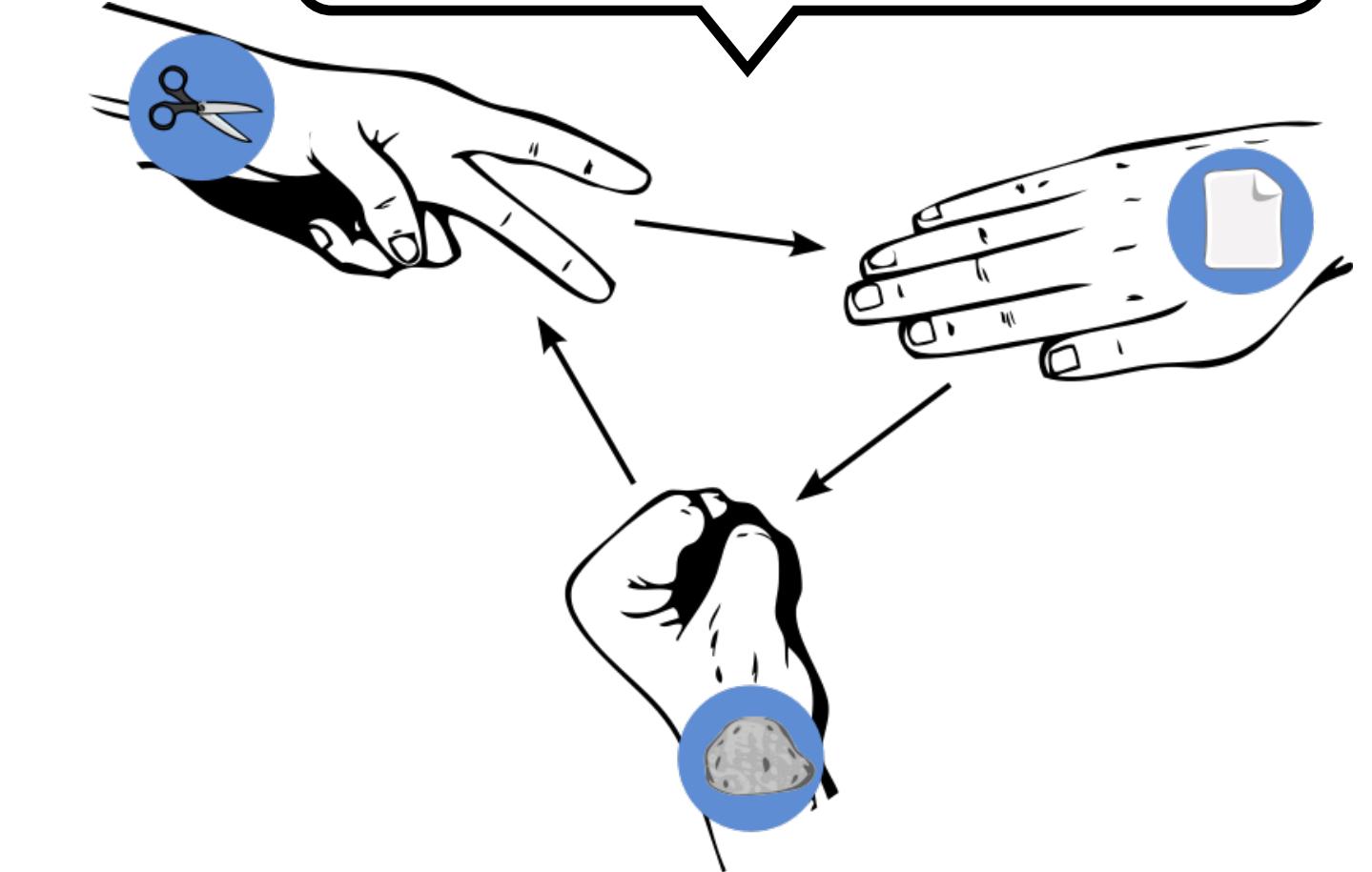


$H(\text{scissors})$

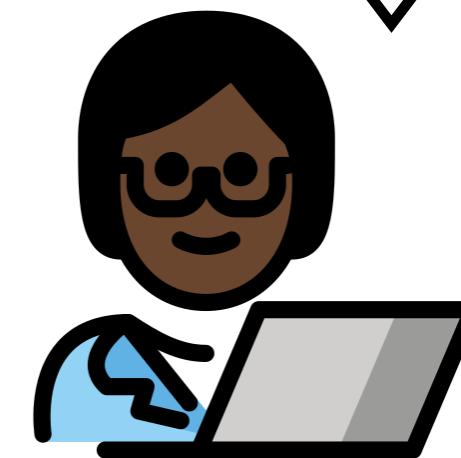
🧐 what's the problem?



rock, paper, scissors



$H(\text{scissors} \parallel r)$



wait until everyone else commits (end of **commit phase**)
and then reveal $(\text{scissors} \parallel r)$ (in the **reveal phase**)

r

A Hash-Based Commitment Scheme

$$\text{Commit}(m, r) = H(m \parallel r) = c$$

🧐 **Binding?** Yes!

$$\Pr[\text{Commit}(m, r) = \text{Commit}(m^*, r^*) \mid m \neq m^* \wedge (m, r, m^*, r^*) \leftarrow \mathcal{A}] \leq \text{negl}(n)$$

$$\Pr[H(m \parallel r) = H(m^* \parallel r^*) \mid m \neq m^*] \leq \text{negl}(|r|)$$

second preimage resistance
of the hash function H

🧐 **Hiding?** Yes!

$$\Pr[b^* = b] \leq \frac{1}{2} + \text{negl}(|r|)$$

$$\Pr[b^* = b \mid m_0, m_1, H(m_b \parallel r)] \leq \text{negl}(|r|)$$

preimage resistance of H

Commitments vs Encryption

🧐 Is it possible to realize a Commitment Scheme using any Encryption Scheme?

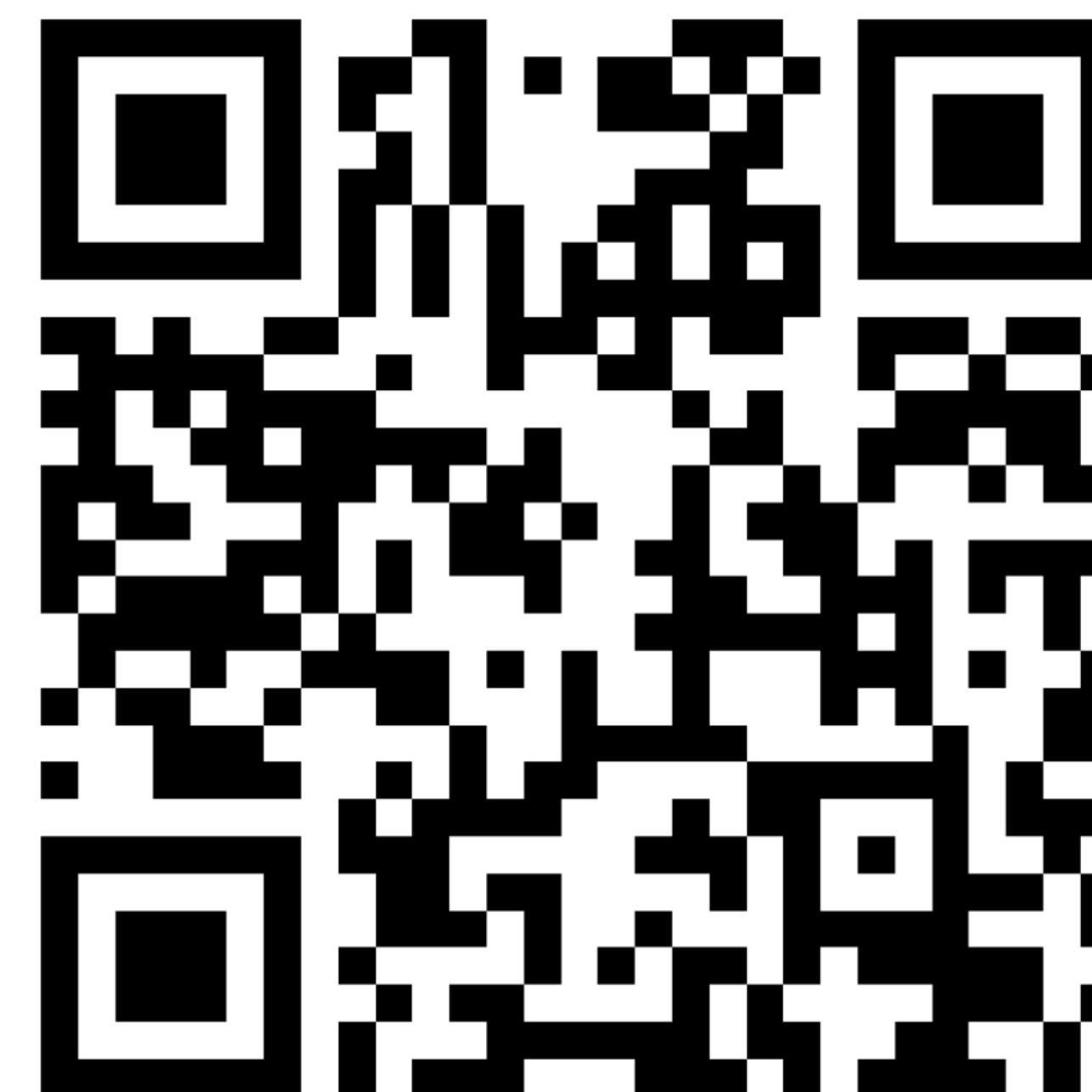
$$\text{Commit}(m, r) = m \oplus r =: c$$

🧐 Hiding?

🧐 Binding?

NOTE: ElGamal encryption gives a good commitment scheme (because the randomness is dLog committed in the cipher text) — there is an exercise in the weekly sheet

ATLQ - Lecture 9



<https://forms.office.com/e/cT5C12rAJR>

Applying cryptography to protect privacy

Applications, usability, and uptake of Privacy Enhancing Technologies

Victor Morel

<https://victor-morel.net/>

Chalmers University of Technology

morelv@chalmers.se

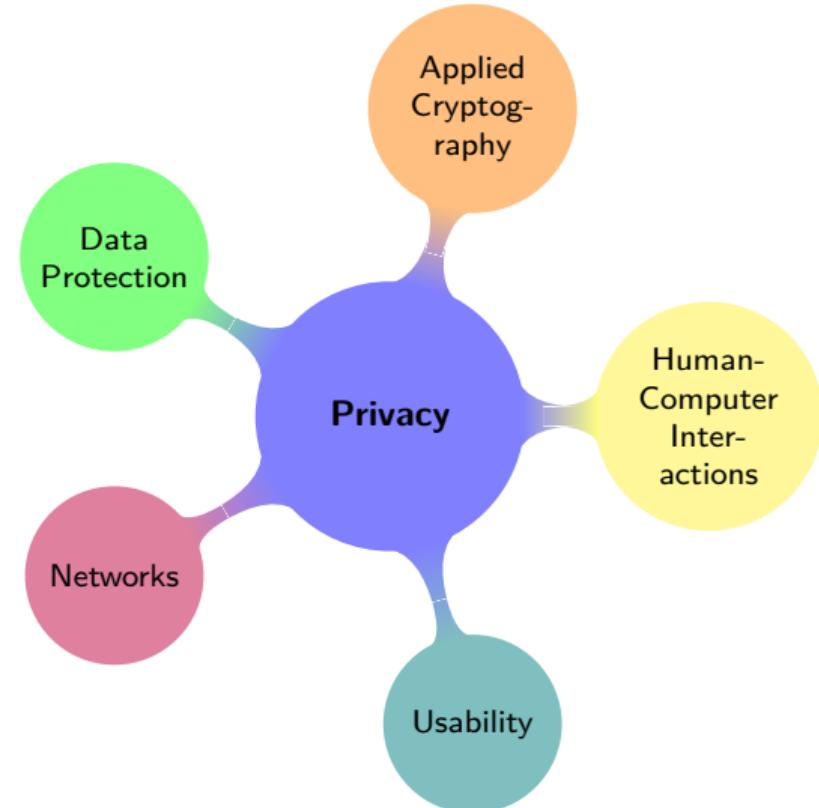


6th December 2024

\$ whoami

Not your usual cryptographer

- › Not a cryptographer at all actually
- › More of a privacy engineer
- › But privacy engineering uses crypto sometimes!
- › Postdoc researcher at iSec



Something happened in 2013



Remember this guy?

GCHQ taps fibre-optic cables for secret access to world's communications

Exclusive: British spy agency collects and stores vast quantities of global email messages, Facebook posts, internet histories and calls, and shares them with NSA, latest documents from Edward Snowden reveal

Boundless Informant: the NSA's secret tool to track global surveillance data

Revealed: The NSA's powerful tool for cataloguing global surveillance data - including figures on US collection

<https://www.theguardian.com/uk/2013/jun/21/gchq-cables-secret-world-communications-nsa>

<https://www.theguardian.com/world/2013/jun/08/nsa-boundless-informant-global-datamining>

Crypto as a remedy?

I will cover

- › Usability of cryptographic tools
- › Deep-dive on one specific Privacy-Enhancing Technology (PET)
- › Zoom-out on the role of cryptography at the scale of society

What I hope you will learn in this lecture

- › Unusable crypto systems are (arguably) bad crypto
- › Outdated crypto defeats its purpose
- › And so does badly implemented and compromised crypto
- › But crypto can be well done! ... and that's really worth it
- › (And also: does crypto have an *ethics*?)

Outline

1 Securing communications

- The bad stuff
- The good stuff

2 Securing data and systems

- The bad stuff
- The good stuff

3 A PET in practice: ABC

- Cryptographic requirements
- Features for an ABC system

- Real world use
- Demonstration
- Yivi

4 Securing society at large

- The bad stuff
- Why the good stuff?
- Ethics of crypto

5 Conclusion

- What crypto cannot do
- Crypto is a tool

Participy - PGP



Pretty Good Privacy

PGP ain't that bad, yet ...

PGP key management sucks

Manual key management is a mug's game. Transparent (or at least *translucent*) key management is the hallmark of every **successful end-to-end secure** encryption system.

Long story short: it involves individuals signing each others' keys

Pitfall

Usability (and therefore uptake)

<https://blog.cryptographyengineering.com/2014/08/13/whats-matter-with-pgp/>
<https://moxie.org/2015/02/24/gpg-and-me.html>

Of the difficulty of having a certificate twenty years ago



Transport Layer Security (TLS)

Web security relies (notably) on SSL/TLS

HTTPS used to be expensive

Not *cryptographically*, but *financially* expensive! In 2002:

GeoTrust sells its certs for \$119 a year, compared to VeriSign's between \$250 and \$350.

Pitfall

Uptake

https://www.theregister.com/2002/07/24/theres_certs_and_certs_verisign/

<https://www.ssldragon.com/blog/evolution-of-ssl-certificates-over-20-years/>

Outline

1 Securing communications

- The bad stuff
- The good stuff

2 Securing data and systems

- The bad stuff
- The good stuff

3 A PET in practice: ABC

- Cryptographic requirements
- Features for an ABC system

- Real world use
- Demonstration
- Yivi

4 Securing society at large

- The bad stuff
- Why the good stuff?
- Ethics of crypto

5 Conclusion

- What crypto cannot do
- Crypto is a tool

Participy - E2E



Out of the box E2E messaging

E2E messaging app by default:



Turnkey encrypted email solutions:



Proton Mail



Uptake

Many communications are secure by default!

Safe communications for whistleblowing



They provide solutions tailored to the *threat level*, as well as pros and cons!

PGP not that bad in combination with Thunderbird, although:

No forward secrecy

Amongst other things

Still, it is fair to conclude that

Usability



Encrypt the web

Google made HTTPS almost mandatory in 2014 (incentive via SEO)

Let's encrypt and *HTTPS Everywhere* finalized the move to encrypt web traffic

HTTPS Is Actually Everywhere

Uptake

 (more than 95% of the global traffic)

HTTPS everywhere in maintenance mode

Outline

1 Securing communications

- The bad stuff
- The good stuff

2 Securing data and systems

- The bad stuff
- The good stuff

3 A PET in practice: ABC

- Cryptographic requirements
- Features for an ABC system

- Real world use
- Demonstration
- Yivi

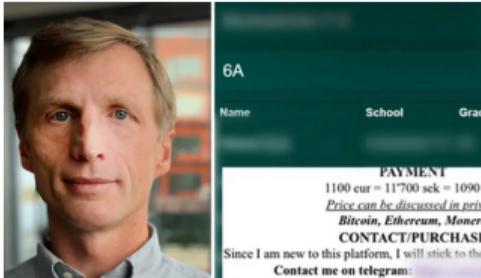
4 Securing society at large

- The bad stuff
- Why the good stuff?
- Ethics of crypto

5 Conclusion

- What crypto cannot do
- Crypto is a tool

Using outdated schemes when storing passwords



Yet another data leak, October 2022

Car maintenance company leaks 12.7k US phone numbers, emails and MD5 unsalted passwords

 Bernard Meyer | Updated on: 28 September 2021 | [Comment](#)

And one with notoriously outdated hash algorithm
(MD5 is breakable on a regular laptop)

Pitfall

Outdated

<https://www.svt.se/nyheter/lokalt/vast/it-experten-om-vklass-och-lackta-elevuppgifterna-i-goteborg>

<https://cybernews.com/security/xado-leaks-us-phone-numbers-emails-md5-unsalted-passwords/>

Participy - Password manager



Bad implementation of password storage schemes



A single point of failure

Pitfall

“A proprietary binary format”

<https://www.wired.com/story/lastpass-breach-vaults-password-managers/>

Outline

1 Securing communications

- The bad stuff
- The good stuff

2 Securing data and systems

- The bad stuff
- The good stuff

3 A PET in practice: ABC

- Cryptographic requirements
- Features for an ABC system

- Real world use
- Demonstration
- Yivi

4 Securing society at large

- The bad stuff
- Why the good stuff?
- Ethics of crypto

5 Conclusion

- What crypto cannot do
- Crypto is a tool

Password manager done well



Usable and regularly audited

Usability & Uptake



<https://bitwarden.com/compliance/>

Outline

1 Securing communications

- The bad stuff
- The good stuff

2 Securing data and systems

- The bad stuff
- The good stuff

3 A PET in practice: ABC

- Cryptographic requirements
- Features for an ABC system

- Real world use
- Demonstration
- Yivi

4 Securing society at large

- The bad stuff
- Why the good stuff?
- Ethics of crypto

5 Conclusion

- What crypto cannot do
- Crypto is a tool

Your ID please!



Alice 🧑 wants to buy 🍷

- SB (verifier) asks Alice (prover) their ID
- But an ID includes lots of superfluous information:
 - Exact age
 - Nationality
 - Gender
 - Name
- Not privacy-friendly!

How can Alice prove she's over 20 without revealing her exact age (nor being tracked)?

Old enough?



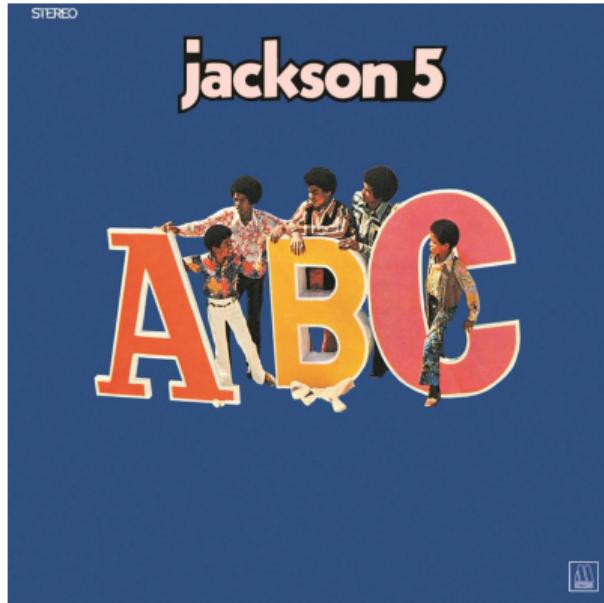
Grow a beard?
→ Sexist and unreliable

A clever (??) trick



Hide the last two digits of your ID card?
→ Only works if you're born in the last century ...

ABC



You mean the song by the Jackson Five?

→ More like Attribute Based Credentials!

ABC? Easy as 1, 2, 3

Briefly put

- › It's a form of authentication mechanism
- › Allows to flexibly and selectively authenticate different attributes about an entity
- › Without revealing additional information about the entity (zero-knowledge property).

For instance

Prove to System Bolaget you are over 20 without disclosing your exact age.

ABC is a system, it requires:

Blocks and properties

- › Basic blocks (some are primitives but not all)
- › Main requirements a protocol should meet

Outline

1 Securing communications

- The bad stuff
- The good stuff

2 Securing data and systems

- The bad stuff
- The good stuff

3 A PET in practice: ABC

- Cryptographic requirements
- Features for an ABC system

- Real world use
- Demonstration
- Yivi

4 Securing society at large

- The bad stuff
- Why the good stuff?
- Ethics of crypto

5 Conclusion

- What crypto cannot do
- Crypto is a tool

Summary of the basic blocks

- › (Generalised) pedersen commitments
- › Cryptographic signatures
- › Group signatures
- › Zero-knowledge proofs
- › Blind signatures

ABC protocols

- › Pedersen commitments
- › Blind signatures
- › ZKP

Demo (and Bonus 3)

Based on group signatures:

- › Cryptographic signatures
- › ZKP

Participy - commitment scheme



Pedersen commitments

Pedersen Commitment Scheme

Tasks 3,4 in HA3

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order q , and h be a random element in $\mathbb{G} \setminus g$.

Let q, g, h be all public information. **Pedersen** commitment function is defined as:

$$\text{Commit}(m, r) = g^m h^r \bmod p = c \quad (\text{for } r \leftarrow \$\mathbb{Z}_q)$$

Binding?

yes, computationally (reduces to DL)

$$\Pr[\text{Commit}(m, r) = \text{Commit}(m^*, r^*) \mid m \neq m^* \wedge (m, r, m^*, r^*) \leftarrow \mathcal{A}] \leq \text{negl}(n)$$

Proof by reduction: \mathcal{A} provides (m, r, m^*, r^*) that yield the same commitment value. The reduction \mathcal{A}' solves the DL instance $(g, h = g^x)$ by sending $x^* = (m^* - m)(r - r^*)^{-1} \bmod q$

Hiding?

yes, information-theoretically (even if the adversary could solve DL)

$$\Pr[b = b^*] = \frac{1}{2} + \frac{1}{q}$$

Proof: show that for any commitment c there exist r_0, r_1 s.t. $c = g^{m_0} h^{r_0} = g^{m_1} h^{r_1}$ (similar to OTP)

13

Credentials signature

- › An Attribute-Based Credential can be realised as a (generalised) Pedersen commitment signed by a credential issuer.
- › Brands¹ proposes a signature that can be used for Attribute-Based Credentials.
- › Camenisch and Lysyanskaya² propose another signature scheme to construct Attribute-Based Credential, also based on a Pedersen commitment.

¹S. A. Brands. Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.

²Camenisch, Jan, and Anna Lysyanskaya. "A signature scheme with efficient protocols." International Conference on Security in Communication Networks.

Group signatures (basically)

Group Signatures



Group signatures (less basically)

Group Signatures

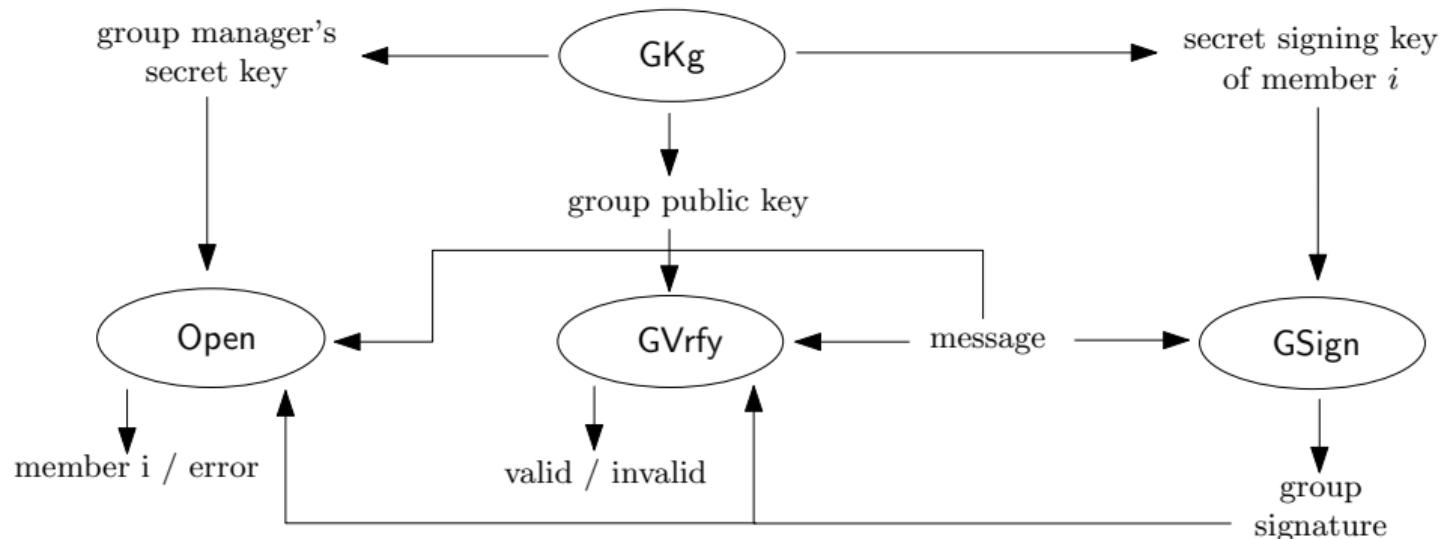


Figure 1.1.: Static Group Signatures

Zero-knowledge proofs

- One party ( the prover) can prove to another party ( the verifier) that a given statement is true while the prover avoids conveying any additional information apart from the fact that the statement is indeed true.
- An (honest verifier) zero-knowledge proof of knowledge has to be:
 - Complete** If the prover knows x , they can convince the verifier
 - Sound** If the prover doesn't know x , they **can't** convince the verifier
 - Zero knowledge** The verifier does not learn any other information but that the prover knows x

Blind signatures



Blind Signatures

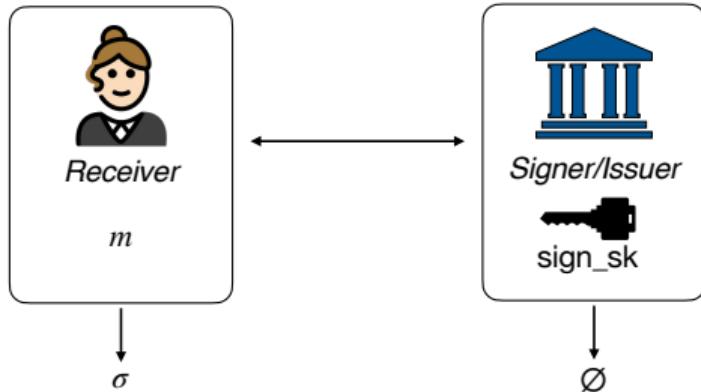
Blind signatures - general view

Definition: Blind Signature

A blind signature scheme is a signature scheme where the signing algorithm algorithms $Sign$ is replaced by an *interactive protocol* run between a signer/issuer (S) and a receiver (R).

The protocol starts with R who has as input a message m , and S who has as input a secret key sk .

At the end of the interaction R obtains a signature σ on m , and S learns nothing about m or σ .

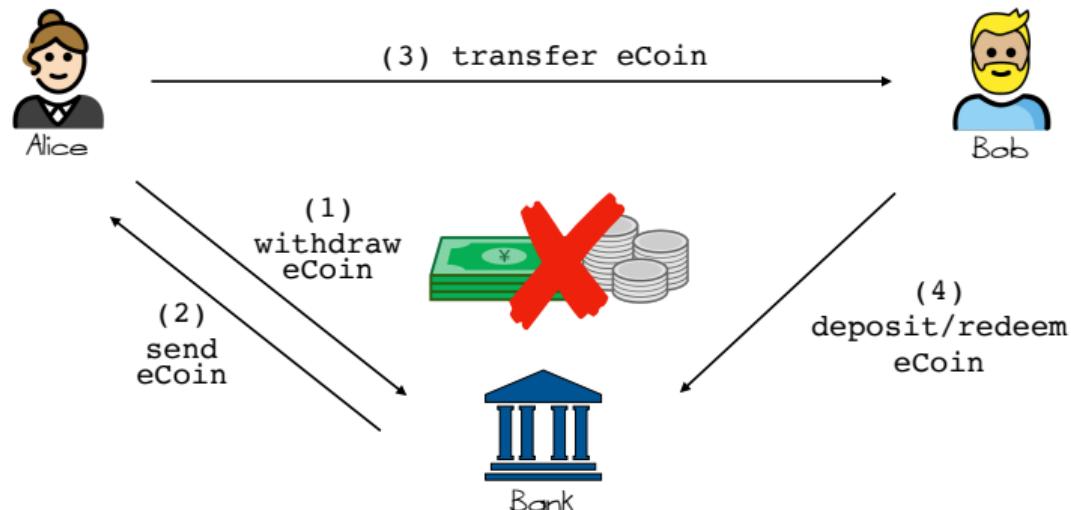


💡 *where can this be useful?*

untraceable electronic payment system
attribute-based credentials [ABC, lecture 12 by Victor]

Blind signatures - a use-case

Chaum's Untraceable eCash System



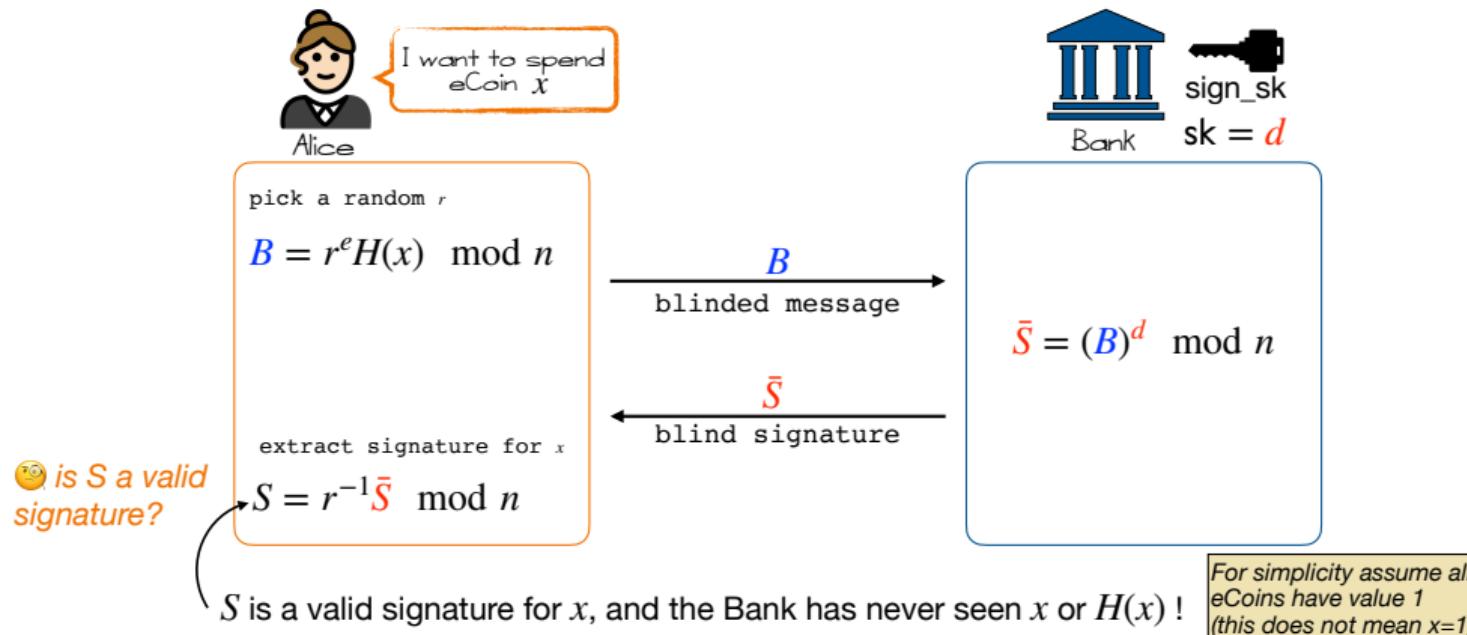
Property Wishlist

1. Only the Bank can generate eCoins
2. Users cannot double spend eCoins (money cloning)
3. eCoins should be untraceable, like physical cash

Blind signatures - a use-case

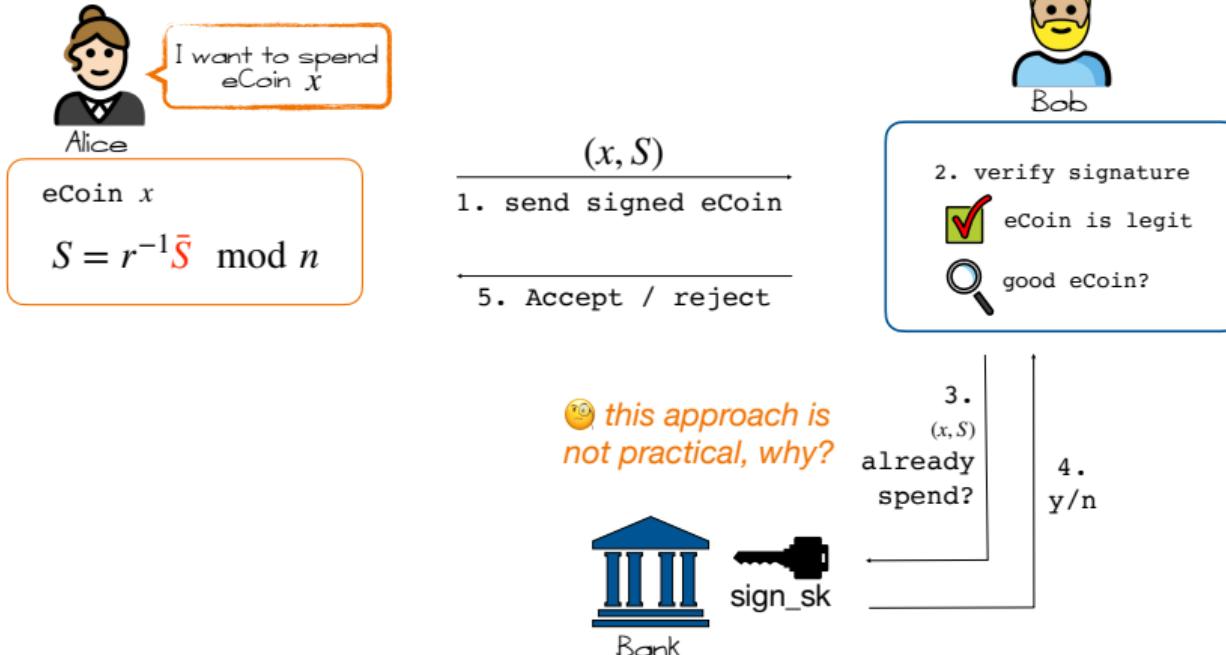
Aim: the Bank should be able to sign an eCoin, **without knowing** what eCoin it is

The eCoin withdrawal procedure with RSA (blind) signatures

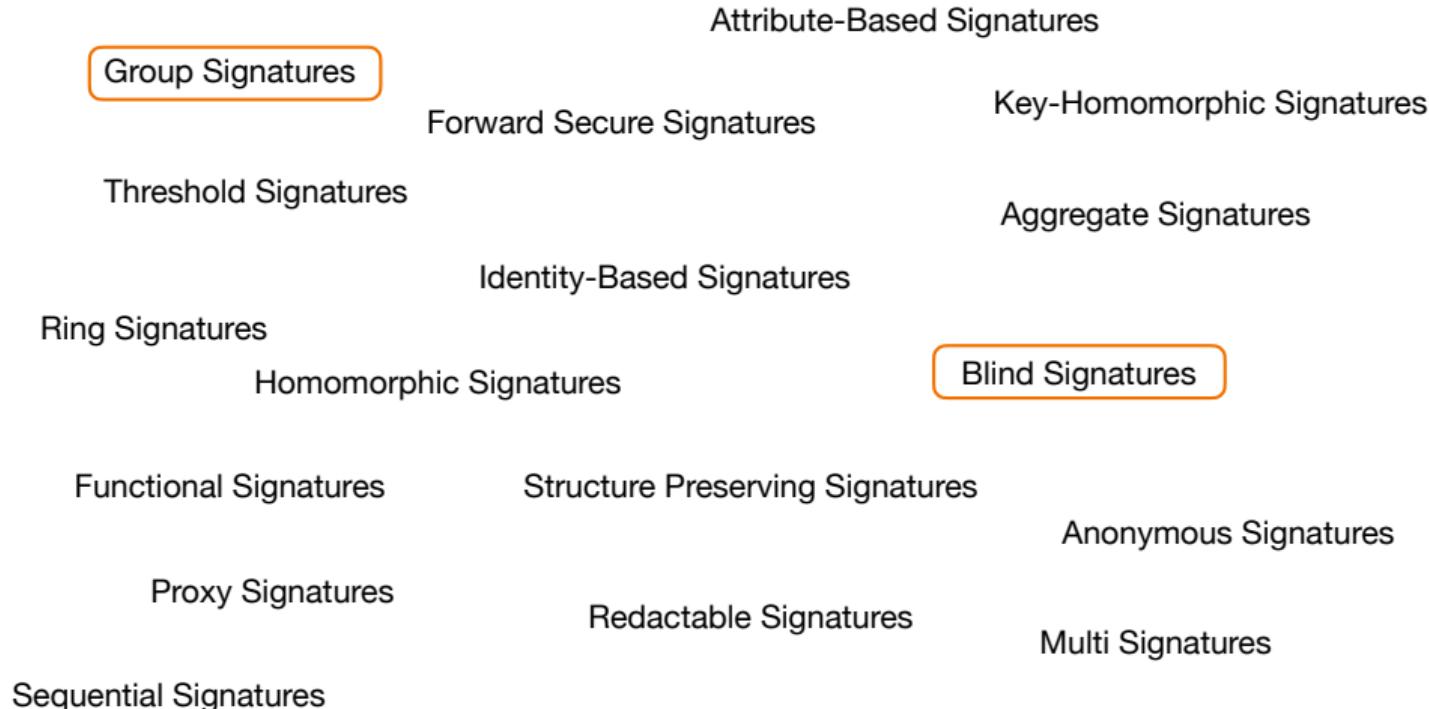


Blind signatures - a use-case

Spending and Redeeming eCoins



Other types of signatures

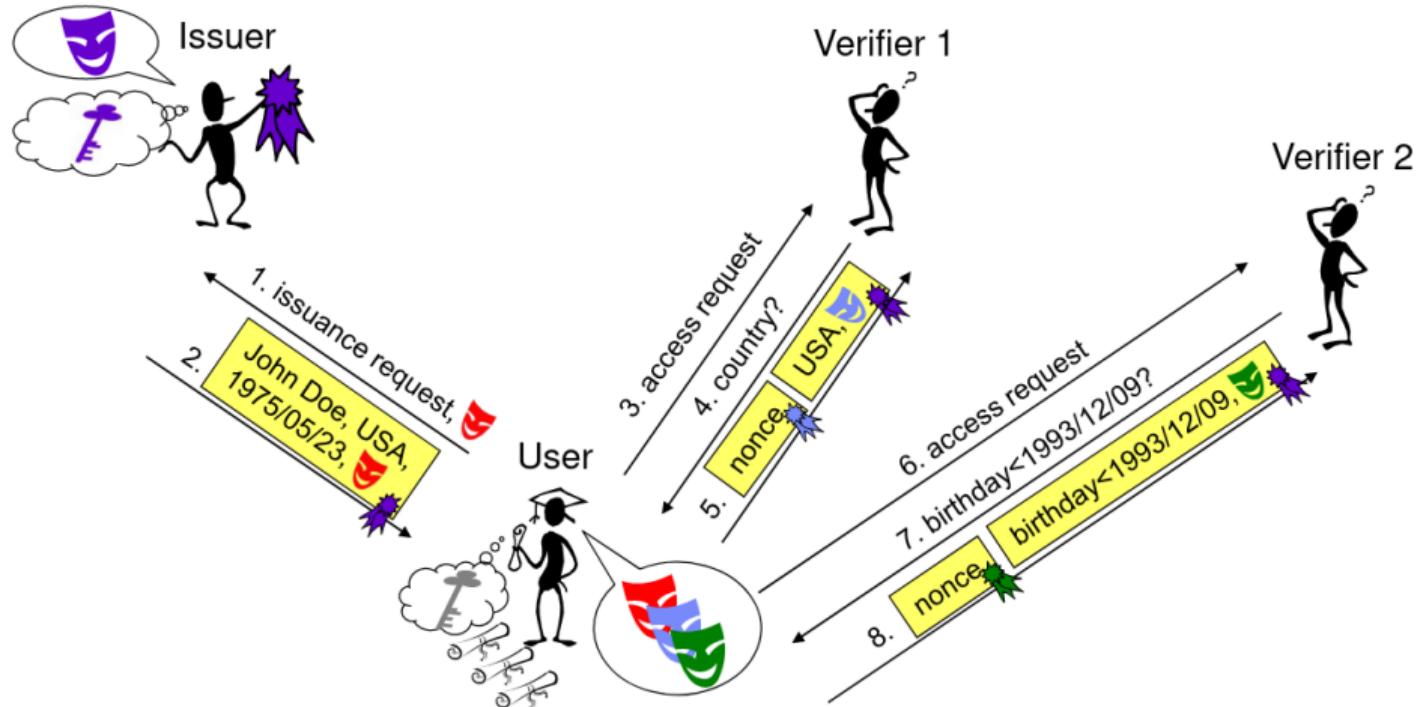


Combining the blocks

Attribute-Based Credentials can be designed as such:

- Construct a suitable signature scheme:
 - a) Consider a commitment scheme;
 - b) Generalize the commitment scheme to a tuple of values instead of only one value;
 - c) Apply a signature on the commitment.
- Develop ABC protocols based on the signature:
 - (a) Use a blinded version of the signature for issuing;
 - (b) Apply a (zero) proof of knowledge for selective disclosure.

A possible implementation - Idemix



<https://csrc.nist.gov/csrc/media/events/meeting-on-privacy-enhancing-cryptography/documents/neven.pdf>

Outline

1 Securing communications

- The bad stuff
- The good stuff

2 Securing data and systems

- The bad stuff
- The good stuff

3 A PET in practice: ABC

- Cryptographic requirements
- Features for an ABC system

- Real world use
- Demonstration
- Yivi

4 Securing society at large

- The bad stuff
- Why the good stuff?
- Ethics of crypto

5 Conclusion

- What crypto cannot do
- Crypto is a tool

Security features (requirements)

- S1 Authenticity: Alice 🧑 has a genuine credential
- S2 Unforgeability: (*3rd* party) Eve 😈 can't forge a credential
- S3 Non-repudiation: Credentials issuer can't deny
- S4 Non-transferability: Alice 🧑 can't transfer her credential to Bob 😊

Privacy features (requirements)

- P1 Offline issuer: The issuer doesn't have to be online when Alice 🚩 buys 🍷
- P2 Issuer unlinkability: The issuer can't track Alice 🚩
- P3 Multi-show unlinkability: System Bolaget 🍷 can't track Alice 🚩
- P4 Selective disclosure: Alice 🚩 can show her age OR her subscription to the 💬💬 independently
- P5 Minimal information: When Alice 🚩 shows her age, her name doesn't leak
→ Although everyone knows Alice here! 😊

Participy - ABC



Outline

1 Securing communications

- The bad stuff
- The good stuff

2 Securing data and systems

- The bad stuff
- The good stuff

3 A PET in practice: ABC

- Cryptographic requirements
- Features for an ABC system

• Real world use

- Demonstration
- Yivi

4 Securing society at large

- The bad stuff
- Why the good stuff?
- Ethics of crypto

5 Conclusion

- What crypto cannot do
- Crypto is a tool

Demonstration

A demonstration of a minimal ABC system

- › Uses group signatures
- › Signatures are not blind
- › Unlinkability is not guaranteed 😢

Demo time! 

Baseline for Bonus HA3

A full-fledged ABC system - Yivi

An open-source application actually used in the real-world!



Yivi - the (privacy-friendly) dutch BankID



Login without password

Your customer no longer needs to
keep complicated password lists, as 1
PIN code is enough to log in with Yivi.



Digital signing and authorisation

Contracts and other electronic documents can be verifiably signed online. And authorise things or actions digitally.

Yivi - usages

In the Netherlands, Yivi is used

- At the Chamber of Commerce
- Healthcare institutions (electronic patient records)
- In municipalities (Amsterdam)
- Universities (SURF)
- Insurances
- etc

Outline

1 Securing communications

- The bad stuff
- The good stuff

2 Securing data and systems

- The bad stuff
- The good stuff

3 A PET in practice: ABC

- Cryptographic requirements
- Features for an ABC system

- Real world use
- Demonstration
- Yivi

4 Securing society at large

- The bad stuff
- Why the good stuff?
- Ethics of crypto

5 Conclusion

- What crypto cannot do
- Crypto is a tool

Backdoors 101

What's the catch?

Encryption backdoors

- › It might mean that something random is not so random
- › Hard to tell if intentional (backdoor) or not (vulnerability)
- › Also related to *front doors*
- Back or front, same result: the system is not safe anymore



<https://www.thesslstore.com/blog/all-about-encryption-backdoors/>

https://www.quora.com/The-NSA-put-a-backdoor-in-Dual_EC_DRBG-Could-there-be-a-backdoor-in-AES-as-well

Surveillance and backdoors

What can go wrong?

Surveillance

- › Illegitimate state-surveillance (NSA, GCHQ, etc)
- › Some governments are pushing for backdoors 
- › ... but it's a very bad idea!
- › Weakening cryptography weakens it for **everybody**

Pitfall

Intentionally compromised crypto

https://www.schneier.com/essays/archives/2007/11/did_nsa_put_a_secret.html

<https://www.theverge.com/2020/10/12/21513212/backdoor-encryption-access-us-canada-australia-new-zealand-uk-india-japan>

<http://dspace.mit.edu/handle/1721.1/97690>

Outline

1 Securing communications

- The bad stuff
- The good stuff

2 Securing data and systems

- The bad stuff
- The good stuff

3 A PET in practice: ABC

- Cryptographic requirements
- Features for an ABC system

- Real world use
- Demonstration
- Yivi

4 Securing society at large

- The bad stuff
- Why the good stuff?
- Ethics of crypto

5 Conclusion

- What crypto cannot do
- Crypto is a tool

Compromised crypto and large-scale surveillance ... so what?

A guy once said:

Arguing that you don't care about the right to privacy because you have nothing to hide is no different than saying you don't care about free speech because you have nothing to say.

Snowden

I don't have anything to hide

... not necessarily for you!

We don't exactly live in an ideal world

- Some socially dominated groups need privacy, it's a human right
- Enshrined in Article 8 of the Charter of Fundamental Rights of the European Union

Why privacy matters (even in an ideal world)

According to Daniel J. Solove:

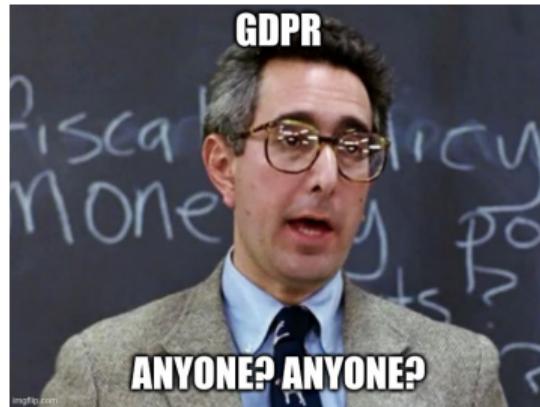
- Limit on Power
- Respect for Individuals
- Reputation Management
- Maintaining Appropriate Social Boundaries
- Trust
- Control Over One's Life
- Freedom of Thought and Speech
- Freedom of Social and Political Activities
- Ability to Change and Have Second Chances
- Not Having to Explain or Justify Oneself

An overlooked link between cryptography and privacy is law.

Encryption in EU law

Encryption enables the application of justice

→ Privacy can be implemented by architecture and by policy
... but the latter cannot function without the former!



Law can foster the use of encryption –
General Data Protection Regulation

- GDPR Art 32(1)(a) talks about *the pseudonymisation and encryption of personal data*;
- DPA can impose fines otherwise (€600,000 for EDF recently)
- Recital 83 encourages the application of security through encryption

Using crypto appropriately

Critical mass (there's safety in numbers)

- › If everyone use it, those who need it are protected: see the use of Tor for instance
- › See also how the widespread use of SSL/TLS prevents eavesdropping and mass surveillance (thanks to the combo *Let's encrypt/HTTPS everywhere*).

Cryptography makes much more sense at the scale of society

- › Standards reduce the odds of encryption backdoors!
- › Large scale adoption can be driven by law (see GDPR)



Outline

1 Securing communications

- The bad stuff
- The good stuff

2 Securing data and systems

- The bad stuff
- The good stuff

3 A PET in practice: ABC

- Cryptographic requirements
- Features for an ABC system

- Real world use
- Demonstration
- Yivi

4 Securing society at large

- The bad stuff
- Why the good stuff?
- Ethics of crypto

5 Conclusion

- What crypto cannot do
- Crypto is a tool

Ethical use-case



Our case study

FBI vs Apple case

- › Yet another shooting attack in the US, culprit killed by the police
- › Police was unable to access the phone because of encryption
- › Apple refused to roll out a *weaker* version of iOS

Other details

- › *May have been another gunman*
- › Create a backdoor for *every* iPhone

Participy - ethics



Ethical, moral, and policy dimensions of cryptography

Maths have moral implications?

Yes: their development (by whom?) and their applications (for what?).

Cryptography rearranges power: it configures who can do what, from what. This makes cryptography an inherently political tool, and it confers on the field an intrinsically moral dimension.

Rogaway, *The Moral Character of Cryptographic Work*

Should private communication be considered a right?

- › If yes, should this right be absolute?
- › What are the implications of such a moral right?

Can we compromise on the absolute character without compromising the underlying mathematical foundations?

Debatable, cf Jaap-Henk Hoepman <https://cutt.ly/7w0YokFB>

Outline

1 Securing communications

- The bad stuff
- The good stuff

2 Securing data and systems

- The bad stuff
- The good stuff

3 A PET in practice: ABC

- Cryptographic requirements
- Features for an ABC system

- Real world use
- Demonstration
- Yivi

4 Securing society at large

- The bad stuff
- Why the good stuff?
- Ethics of crypto

5 Conclusion

- What crypto cannot do
- Crypto is a tool

Cannot protect against business surveillance



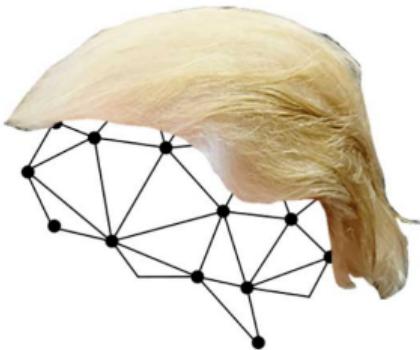
The security of the lock doesn't matter if you give away the keys



degooglisons-internet.org

<https://www.theguardian.com/books/2019/oct/04/shoshana-zuboff-surveillance-capitalism-assault-human-autonomy-digital-privacy>

State and corporate surveillance are two sides of the same coin



Cambridge
Analytica



<https://www.cnil.fr/en/use-google-analytics-and-data-transfers-united-states-cnil-orders-website-manageroperator-comply>

Outline

1 Securing communications

- The bad stuff
- The good stuff

2 Securing data and systems

- The bad stuff
- The good stuff

3 A PET in practice: ABC

- Cryptographic requirements
- Features for an ABC system

- Real world use
- Demonstration
- Yivi

4 Securing society at large

- The bad stuff
- Why the good stuff?
- Ethics of crypto

5 Conclusion

- What crypto cannot do
- **Crypto is a tool**

A tool to protect privacy



Crypto is a tool



One needs to know how to handle a tool

Crypto is not the only tool to protect privacy

- › Law is a tool as well
- › Privacy by architecture and by policy go hand in hand

Recap From the Last Lectures

Secret Sharing

Additive

Replicated

Shamir

Polynomials
 \mathbb{Z}_p

Lagrange
Interpolation

Commitments

hiding

binding

PROOF

Pedersen

Hash-based

**Privacy
Enhancing
Technology
(PET)**

RSA blind signature

Attribute Based Credentials (ABC)

**Threshold
Cryptography
(ElGamal)**

Lecture Agenda

Commitment Schemes

- Commitments VS Encryption
- Impossibility Result - Proof

Verifiable Secret Sharing (VSS)

- A Simple Scheme
- Shamir+Pedersen

Zero-Knowledge Proofs (ZK)

- Introduction
- Schnorr Protocol
- Proving Knowledge of Pedersen

Removing Interaction (NIZK)

- Fiat-Shamir Heuristic

Commitments vs Encryption

🧐 Is it possible to realize a Commitment Scheme using any Encryption Scheme?

$$\text{Commit}(m, r) = m \oplus r =: c$$

🧐 Hiding?

🧐 Binding?

NOTE: ElGamal encryption gives a good commitment scheme (because the randomness is dLog committed in the cipher text) — there is an exercise in the weekly sheet

An (Important) Security Note

Theorem (impossibility) No commitment scheme can be information-theoretically binding *and* information-theoretically hiding at the same time.

Proof. (by reduction to absurd)

Suppose we have a scheme which is both information-theoretically hiding and binding, (now we want to conclude that this contradicts the statement of the theorem) and suppose the committing party (the sender) generates a commitment $c = \text{Commit}(m, r)$. The information-theoretical hiding property implies that there must exist values m^* ($\neq m$) and some randomness r^* such that $c = \text{Commit}(m^*, r^*)$; why? otherwise an infinitely powerful receiver could break the concealing property (find the unique pair (m, r) that generates the commitment value c). But now we know that there exists an m^* s.t. $\text{Commit}(m^*, r^*) = c = \text{Commit}(m, r)$ which means the commitment is not binding (and an infinitely powerful sender can find such a collision m^*). So we conclude that if the commitment scheme is information-theoretically hiding it cannot be binding for a computationally unbounded sender, which contradicts the statement of the theorem. The conclusion is that it was absurd to assume the existence of a commitment scheme that is both hiding and binding in a information-theoretically way.

Lecture Agenda

Commitment Schemes

- Commitments VS Encryption
- Impossibility Result - Proof

Verifiable Secret Sharing (VSS)

- A Simple Scheme
- Shamir+Pedersen

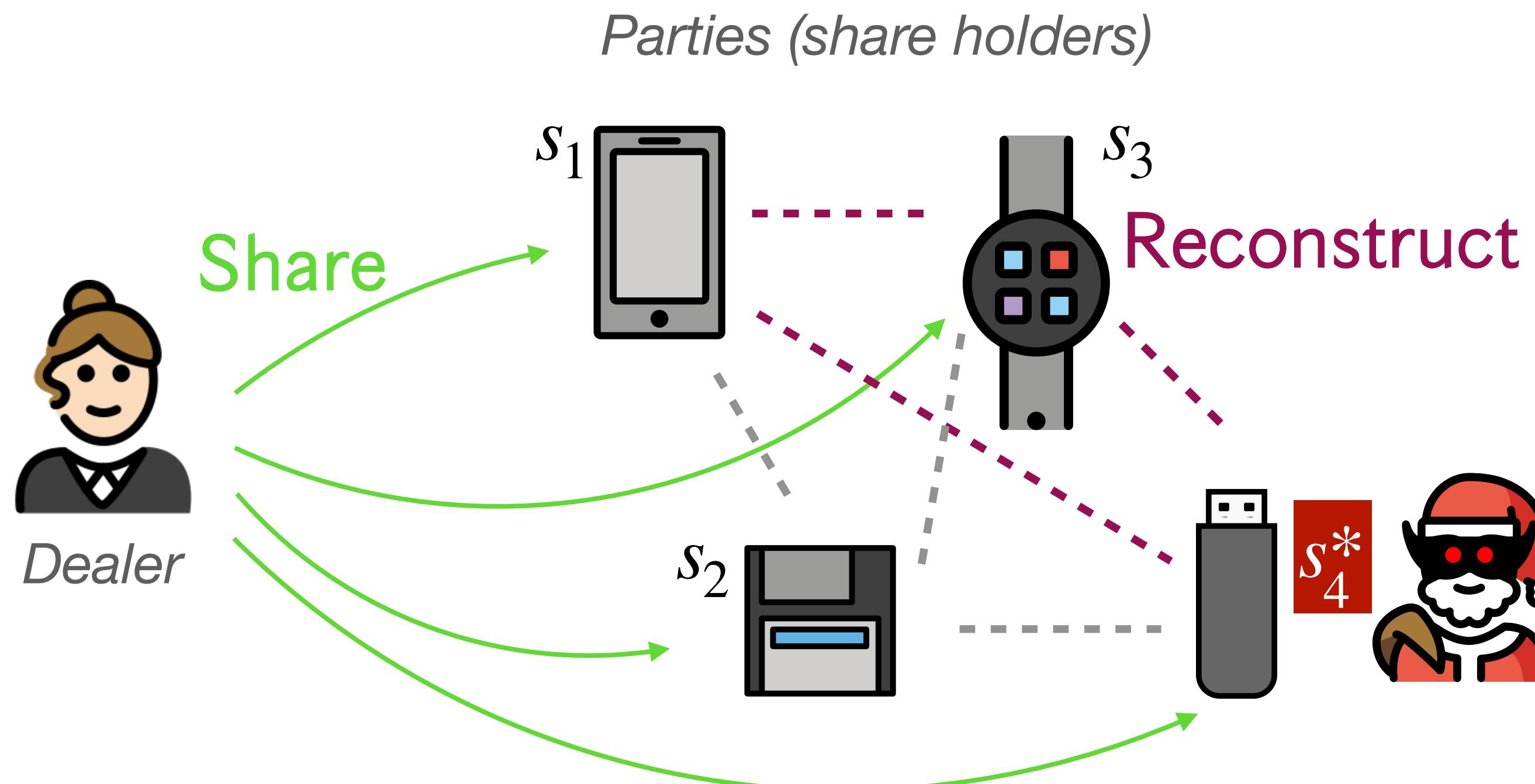
Zero-Knowledge Proofs (ZK)

- Introduction
- Schnorr Protocol
- Proving Knowledge of Pedersen

Removing Interaction (NIZK)

- Fiat-Shamir Heuristic

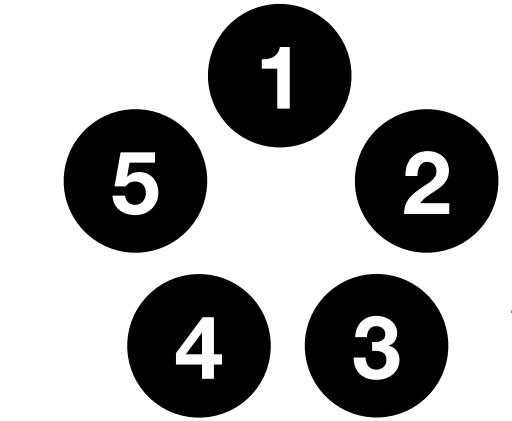
What Is the Problem?



A “Simple” Verifiable Secret Sharing (VSS) Scheme

Take any **threshold** secret sharing scheme (e.g. Shamir)

Assuming you have only 1 **corrupt share**, among the n shares



$$n = 5, t = 2, \binom{n}{t} = 10$$

Run Reconstruct on **all possible sets of shares** of size t

⚠️ *how many distinct sets of size t are there in $\{1, 2, \dots, n\}$?*

Verify the secret by checking **the most common** s among all the reconstructed values

⚠️ *This works but it has limitations ...*

Pedersen Verifiable Secret Sharing (VSS)

- **Share(s):** [this algorithm is run by the dealer, who holds the secret $s \in \mathbb{Z}_p$]

Sample at random $a_1, \dots, a_{t-1}, b_0, \dots, b_{t-1} \leftarrow \mathbb{Z}_p$ ($a_{t-1} \neq 0 \neq b_{t-1}$)

Define the polynomials: $a(x) = s + a_1x + \dots + a_{t-1}x^{t-1}$

$$b(x) = b_0 + b_1x + \dots + b_{t-1}x^{t-1}$$

Compute the share for party P_i as: $s_i = (a(i), b(i))$ [s_i is sent to P_i via a secure channel]

Publish the set $\{C_i\}_{i=0}^{t-1}$ of commitments to each share: $C_i = g^{a_i}h^{b_i}$, where $a_0 = s$

- **Verify($s_i, \{C_j\}$):** Check share consistency $g^{s_i[1]} \cdot h^{s_i[2]} = = \prod_{j=0}^{t-1} (C_j)^{i^j}$

- **Reconstruct($\{s_{i_1}, \dots, s_{i_t}\}, \{C_{i_1}, \dots, C_{i_t}\}$):** do like in Shamir secret sharing to reconstruct $a(0) = s$ and $b(0) = b_0$ [Lagrange interpolation with evaluation at 0] and verify the consistency $g^s \cdot h^{b_0} = = C_0$

Lecture Agenda

Commitment Schemes

- Commitments VS Encryption
- Impossibility Result - Proof

Verifiable Secret Sharing (VSS)

- A Simple Scheme
- Shamir+Pedersen

Zero-Knowledge Proofs (ZK)

- Introduction
- Schnorr Protocol
- Proving Knowledge of Pedersen

Nollkunskapsbevis

Null-Wissen-Beweis

Prueba de conocimiento cero

Dimostrazione a conoscenza zero

Preuve à divulgation nulle de connaissance

零知识证明

Removing Interaction (NIZK)

- Fiat-Shamir Heuristic

Interactive Proofs

You can't have your cake and eat it too!



well, with crypto you do :)

zero knowledge proofs

<https://youtu.be/Opu8tvH2sc?t=68>

How To Formalise This Into Math/Crypto?

5	3			7				
6			1	9	5			
	9	8				6		
8			6				3	
4		8	3				1	
7			2			6		
6				2	8			
	4	1	9				5	
	8			7	9			

x

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

w

set of all valid
sudoku starters

solution satisfies
sudoku rules

The most general formalisation: $x \in L \text{ iff } \exists w \text{ s.t. } R(x, w) = 1$

How To Formalise This Into Math/Crypto?

Definition: A **zero-knowledge (ZK) proof** is an *interactive protocol*, run between two parties called Prover (P) and Verifier (V), in which the Prover *probabilistically* convinces the Verifier of the correctness of a given mathematical proposition, and the Verifier learns *nothing beyond the truth of the statement*.

	Relation $R(x; w) = 1$ iff	statement (x)	witness (w)
dLog:	$x = g^w \in \mathbb{G}$	pk	sk
factoring:	$x = w[1]w[2] \wedge w[1], w[2]$ are primes	N	(p, q)

We will work only work with a special case of ZK protocols called sigma protocols (Σ -protocols)

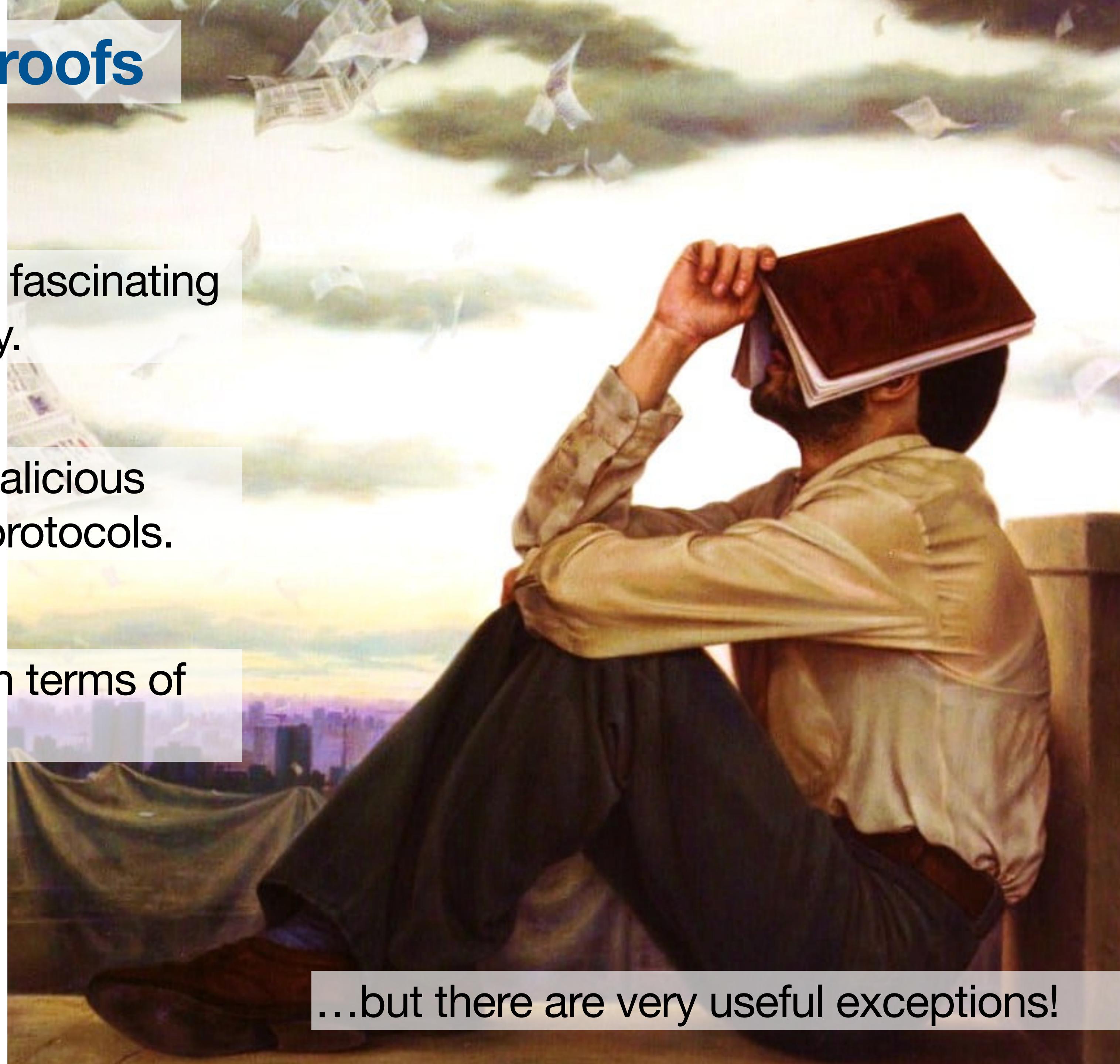
A Few Words About ZK Proofs

The theory of ZK Proofs is extremely fascinating and it is fundamental in cryptography.

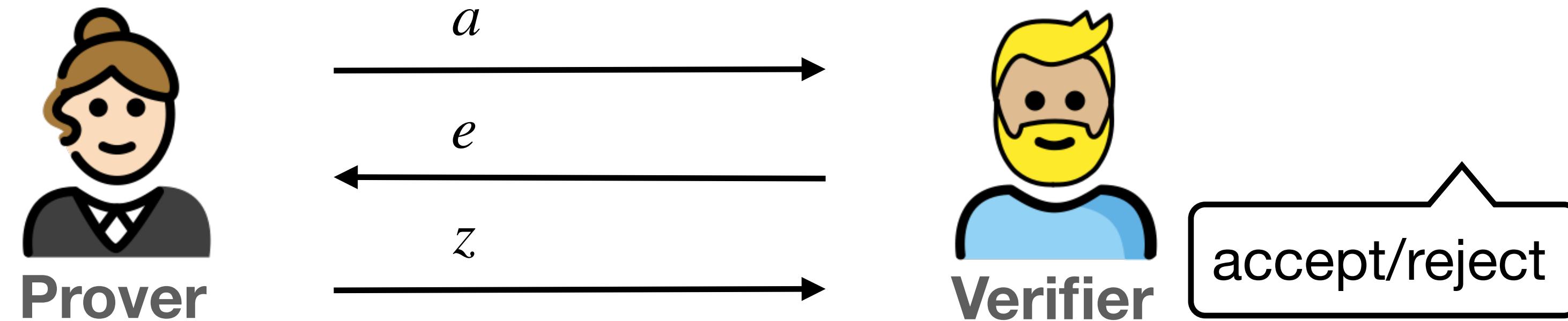
ZK Proofs can be used to achieve malicious security in multi-party computation protocols.

In general ZK Proofs are expensive in terms of computation & communication.

...but there are very useful exceptions!



(Sigma) Σ -Protocols



Definition: A **Σ -protocol** for relation R is a *three-move, public-coin protocol* of the form depicted above that satisfies the following three properties:

- **Completeness:** If P and V follow the protocol on input x and private input w to P where $(x, w) \in R$, then V always accepts.
- **Special soundness:** There exists a PPT algorithm \mathcal{E} (extractor) that given any x and any pair of accepting transcripts $(a, e, z), (a, e', z')$ for x , with $e \neq e'$, outputs w such that $(x, w) \in R$.
- **Special honest verifier zero knowledge (SHVZK):** for every x and w such that $(x, w) \in R$ and every $e \in \{0,1\}^t$ there exists a PPT algorithm Sim (simulator) which given in input (x, e) outputs transcripts $(a; e; z)$ that are distributed like real conversations:

$$\{\text{Sim}(x, e)\} =_c \{\langle P(x, w), V(x, e) \rangle\}$$

Intuition of the Properties of Σ -Protocols

Completeness

The protocol “completes” (ends) correctly, i.e., the verifier will always accept an honest prover

Special Soundness

If the verifier accepts it must be the case that the prover knows the witness (with large enough probability) — This property protects the verifier

Special Honest Verifier Zero Knowledge

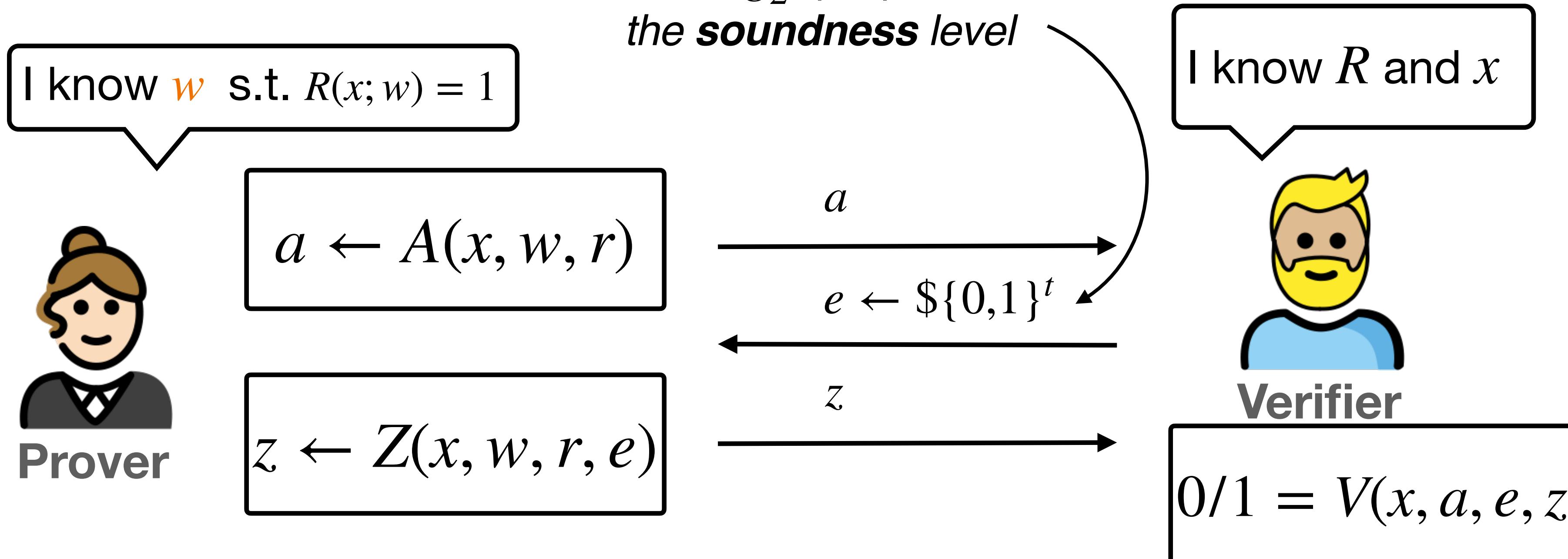
A transcript does not leak the witness (since the transcript could be efficiently simulated) — This property protects the prover

Remarks

It is in the prover's interest not to choose the same r (thus a) for two protocol runs. But the fact that from two distinct conversations that start with the same a we can mathematically extract w ensures us that only a prover who knows w can answer correctly to any challenge.

It is in the verifier's interest to choose the challenge e at random, and after receiving a from the prover. But the fact that knowing e in advance we have a mathematical way to simulate accepting conversations ensures that a transcript transfers “zero knowledge” about the witness.

Generic Form of a Σ -Protocol (A,Z,V)



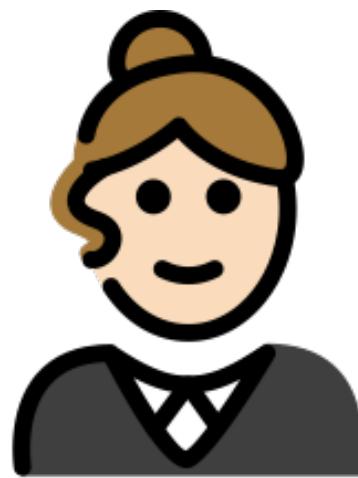
Schnorr Σ -Protocol (Aka Schnorr Identification)

useful for Task 3 in HA3
special soundness

Proving knowledge of the **secret key** corresponding to a public key of the form $h = g^w$

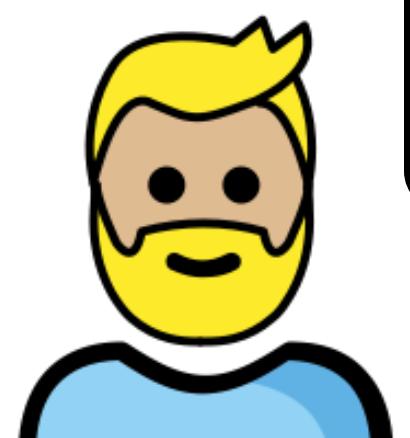
where $\mathbb{G} = \langle g \rangle$ is a group of large prime order q

Yes, I know w s.t. $h = g^w$!

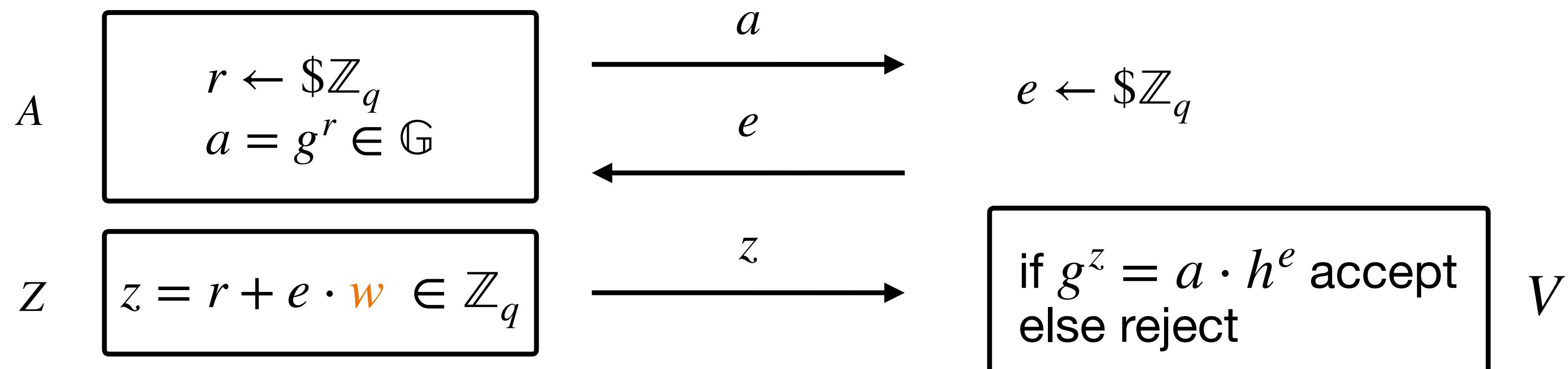


Prover

Are you the holder of the secret key corresponding to $\text{pk} = h$?



Verifier



Given $(a; e; z), (a; e'; z')$ extract w
 $w = (z - z') \cdot (e - e')^{-1} \pmod q$

Completeness
Special Soundness
SHVZK

Given e generate a transcript with correct distribution

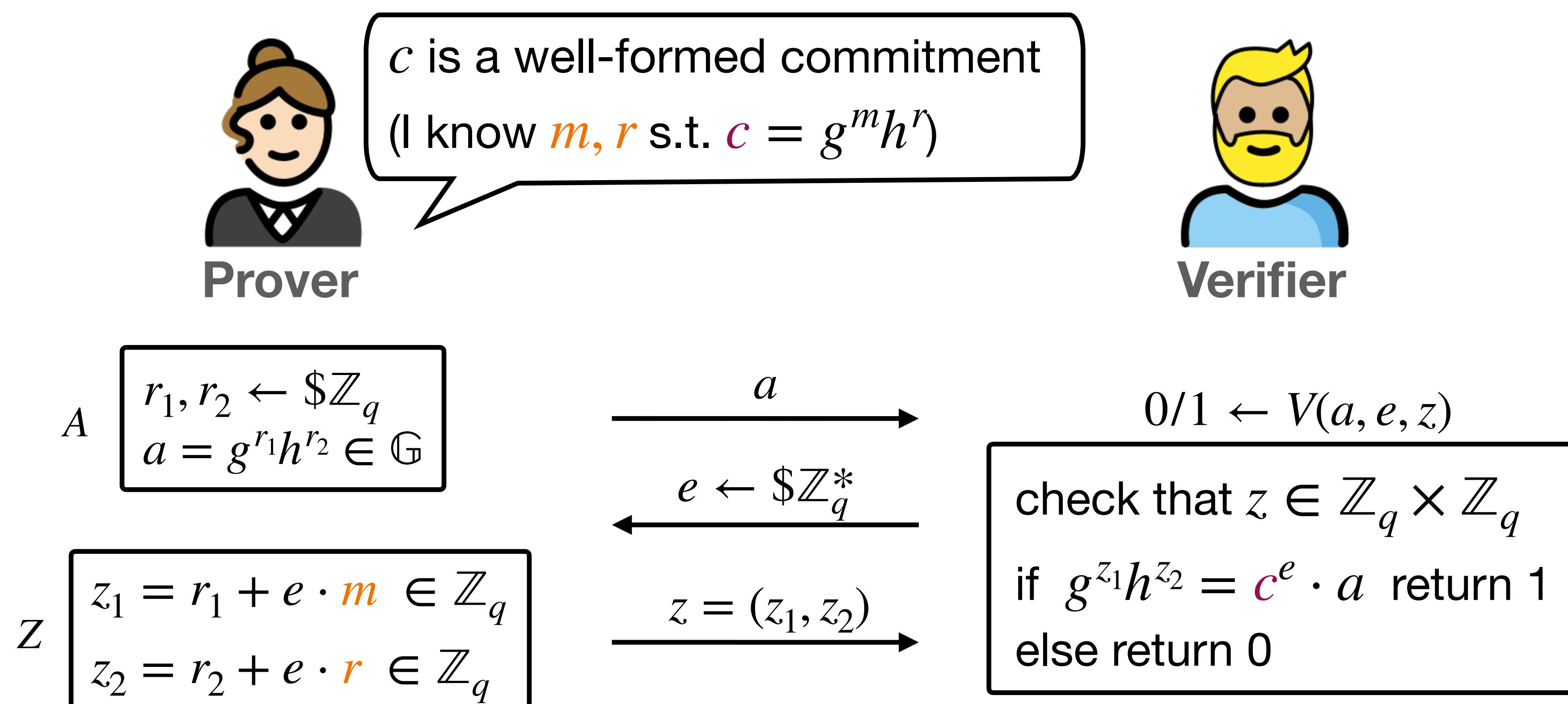
$$\{(a; e; z) : r \leftarrow \mathbb{Z}_q; a = g^r; z = r + ew\}$$
$$\{(a; e; z) : z \leftarrow \mathbb{Z}_q; a = g^z h^{-e}\}$$

Proving Knowledge of Pedersen Commitments

useful for Task 4 in HA3

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of (large) prime order q , and h be a random element in $\mathbb{G} \setminus \{g\}$. Let q, g, h be public information. The **Pedersen** commitment function is defined as:

$$\text{Commit}(m, r) = g^m h^r = c \quad (\text{for } r \leftarrow \mathbb{Z}_q)$$



Lecture Agenda

Commitment Schemes

- Commitments VS Encryption
- Impossibility Result - Proof

Verifiable Secret Sharing (VSS)

- A Simple Scheme
- Shamir+Pedersen

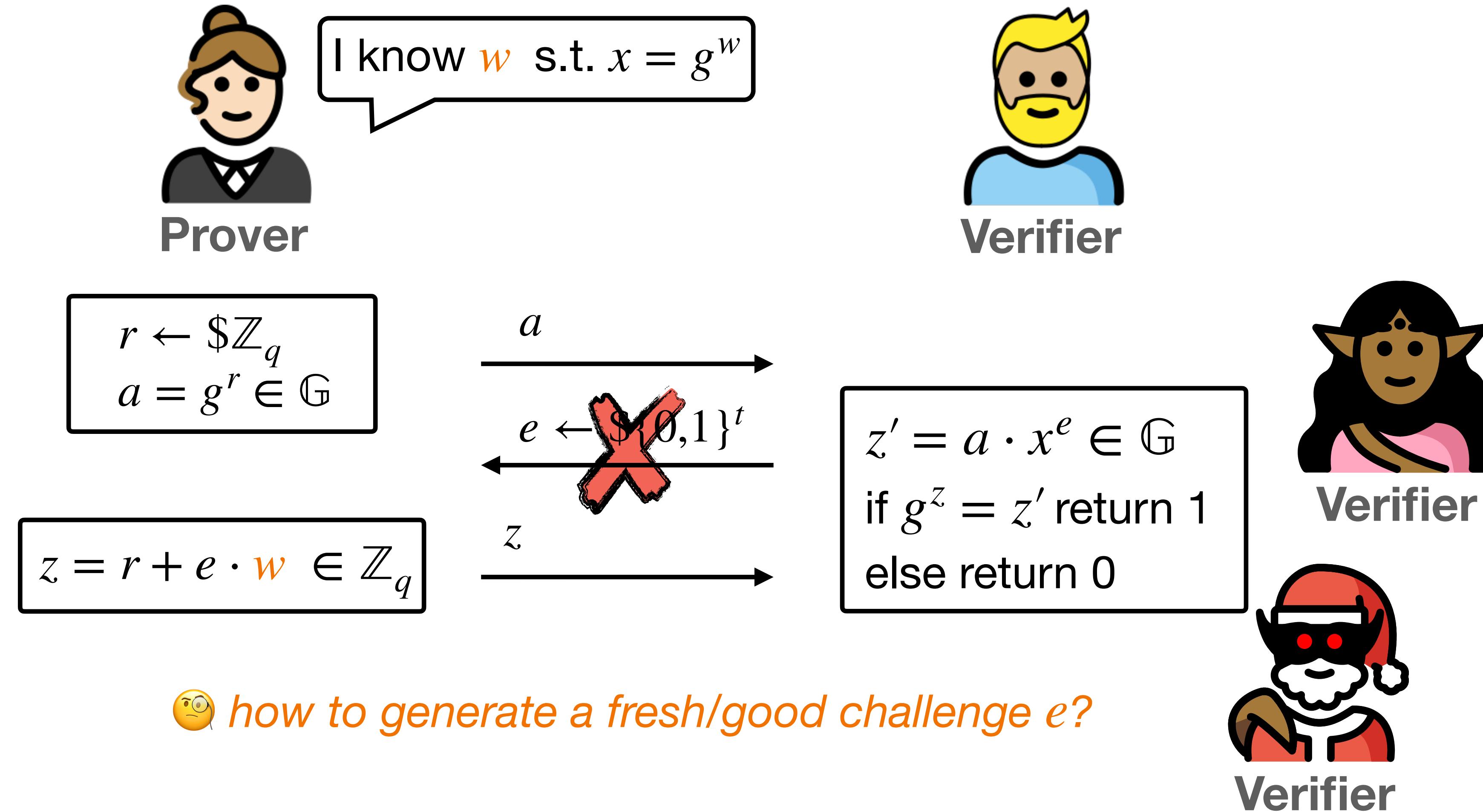
Zero-Knowledge Proofs (ZK)

- Introduction
- Schnorr Protocol
- Proving Knowledge of Pedersen

Removing Interaction (NIZK)

- Fiat-Shamir Heuristic

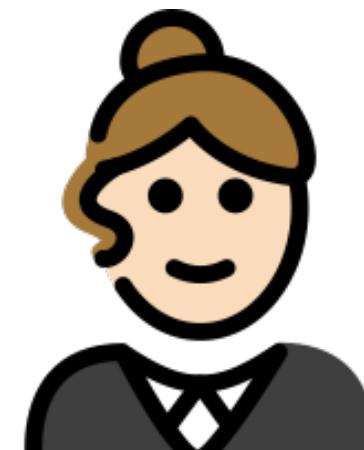
Schnorr Σ -Protocol ... NON-Interactive?



Fiat-Shamir Heuristic compute the challenge as $e = H(g, x, a)$
(and model the hash function as a random oracle)

Schnorr Σ -Protocol ... NON-Interactive?

useful for Task 5 in HA3



I know w s.t. $x = g^w$

Prover

$$r \leftarrow \mathbb{Z}_q$$
$$a = g^r \in \mathbb{G}$$

$$e = H(g, x, a)$$

$$z = r + e \cdot w \in \mathbb{Z}_q$$

Recipe to create a digital signature from a ZK Proof

1. pick randomness
2. generate new (unpredictable) randomness using the hash function
3. use the secret and hide it with both of the randomnesses
4. return a proof of knowledge of the secret value

🤔 where is the message?

This is the same procedure as ECDSA is built

$$s = \text{inv}(k) \cdot (z + d \cdot r) \bmod n$$

Fiat-Shamir Heuristic compute the challenge as $e = H(g, x, a)$
(and model the hash function as a random oracle)

HA3 First Hand-in Deadline Is Next Tuesday

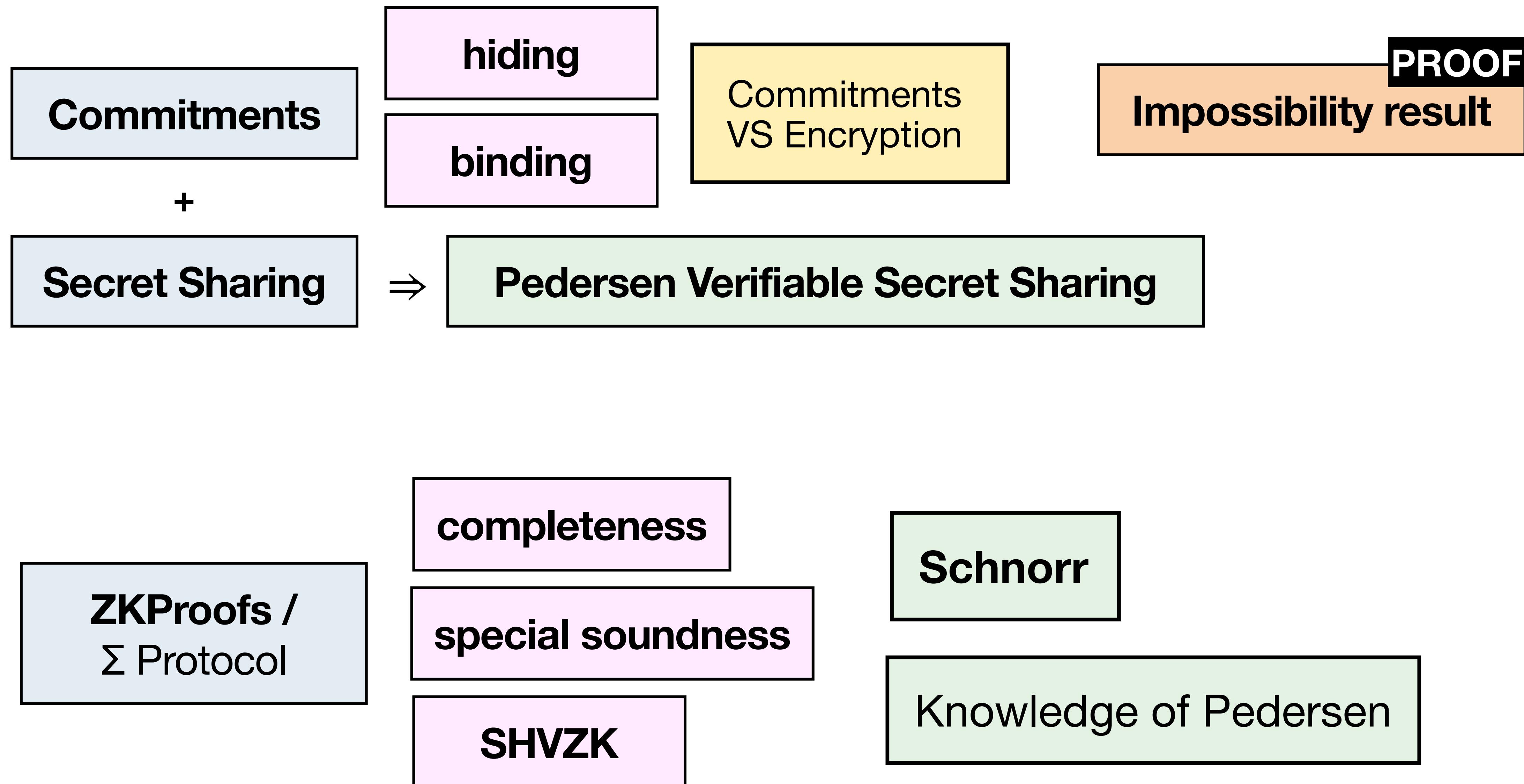
Last Office Hours on Monday 11:30-12:30

Final Deadline for HAs: Jan 8th (FIRM)

Next Tuesday: “About the Exam” + Q&A

Exam: Jan 17th (Morning)

Recap From the Last Lecture



Lecture Agenda

Composing Σ Protocols

- In Parallel
- AND Proofs
- OR Proofs [Not in Exam]

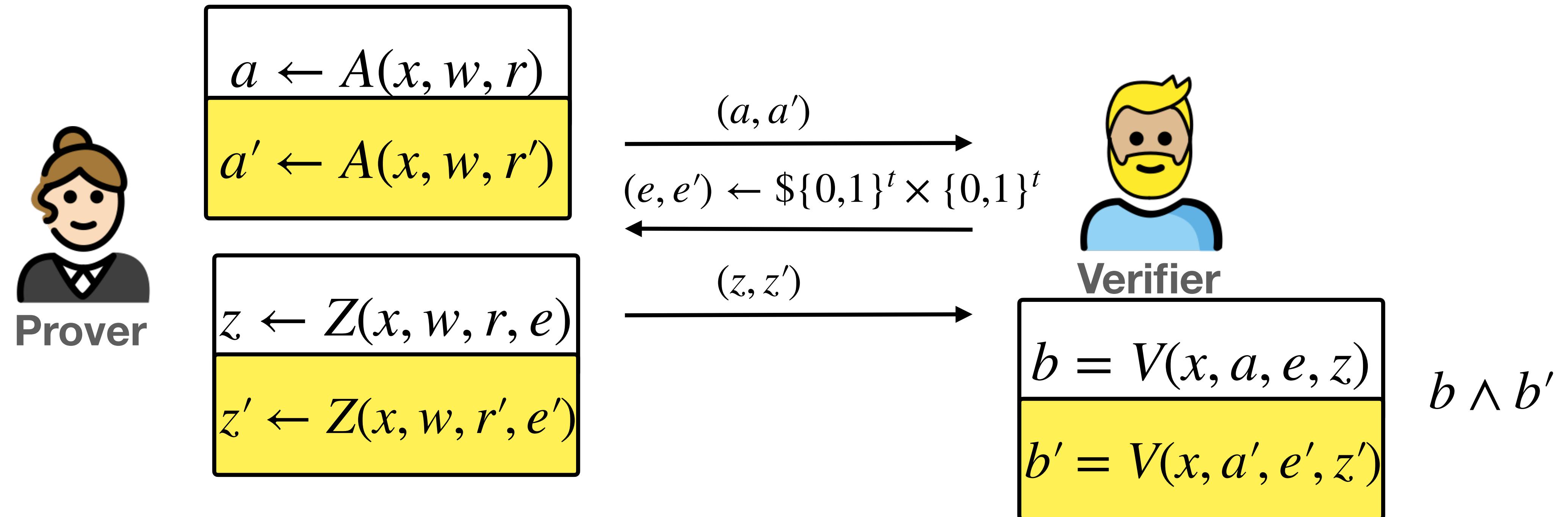
Private Information Retrieval (PIR)

- Problem Statement
- The Square Root PIR

Post Quantum Cryptography (PQC)

- The Lifespan of a Cryptosystem
- Classical VS Quantum Computers
- The Landscape of PQC
- Lamport Signature

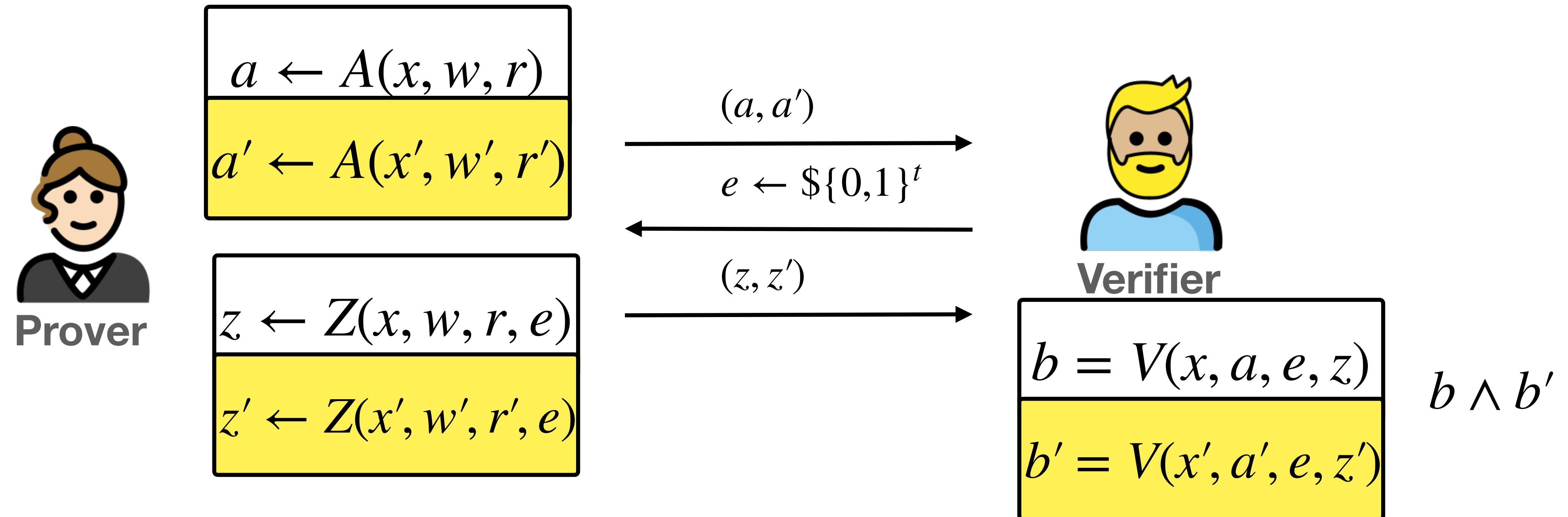
Parallel Repetition of Σ -Protocols



🧐 what is this useful for?

Amplify the soundness level of the protocol (same statement x and witness w , longer challenge)

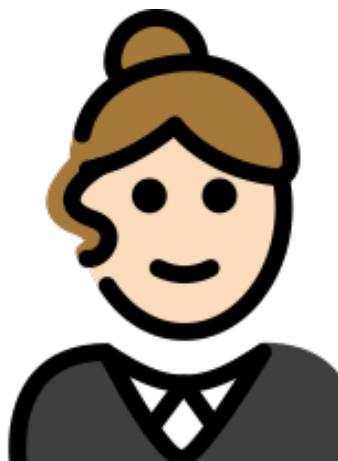
AND Composition



make P prove both statements in parallel using a *single challenge* e for both proofs.

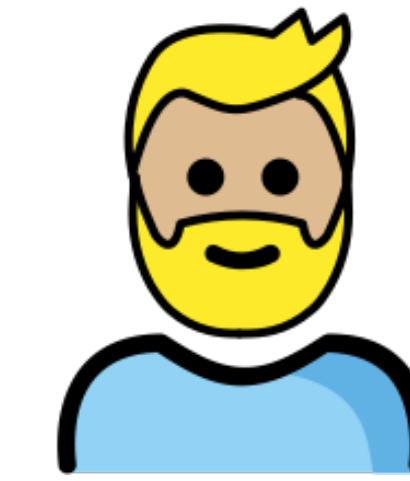
An Example: Chaum–Pedersen Σ -Protocol (Proof of Same dLog)

$$R = \{ ((\mathbb{G}, q, g, h, x_1, x_2), w) \mid g, h \in \mathbb{G} \text{ & } x_1 = g^w \text{ & } x_2 = h^w \}$$



Prover

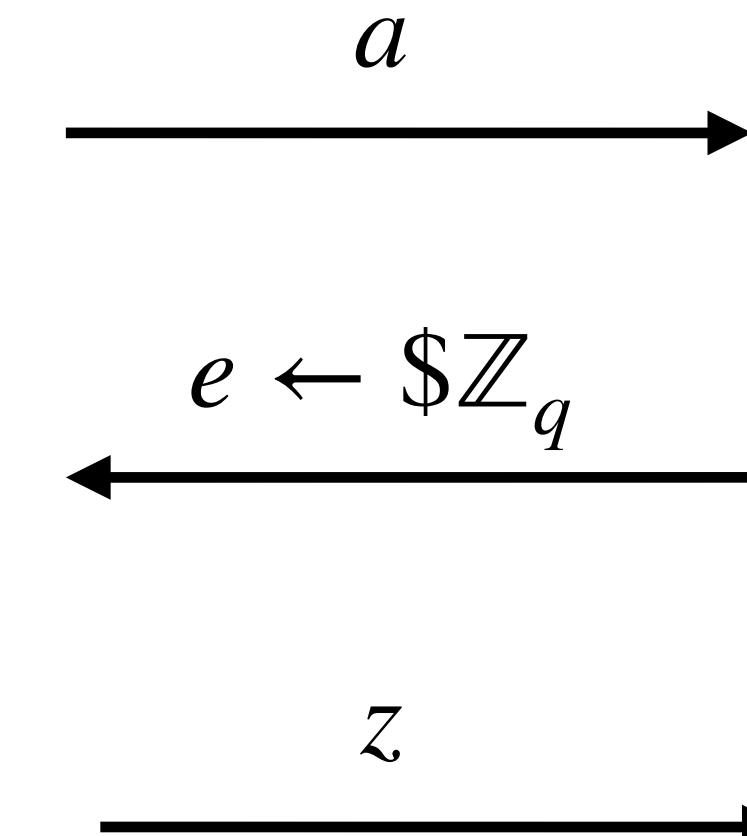
$ZKPoK\{w : x_1 = g^w \text{ and } x_2 = h^w\}$
zero-knowledge proof of knowledge



Verifier

$r \leftarrow \$\mathbb{Z}_q$
 $a_1 = g^r \in \mathbb{G}$
 $a_2 = h^r \in \mathbb{G}$
 $a = (a_1, a_2) \in \mathbb{G}^2$

$z = r + e \cdot w \in \mathbb{Z}_q$



if $g^z = a_1 x_1^e \wedge h^z = a_2 x_2^e$
return 1
else return 0

🧐 *Looks familiar?* This solution is almost a parallel repetition of Schnorr
(except that the challenge(s) are now squashed into a single one)

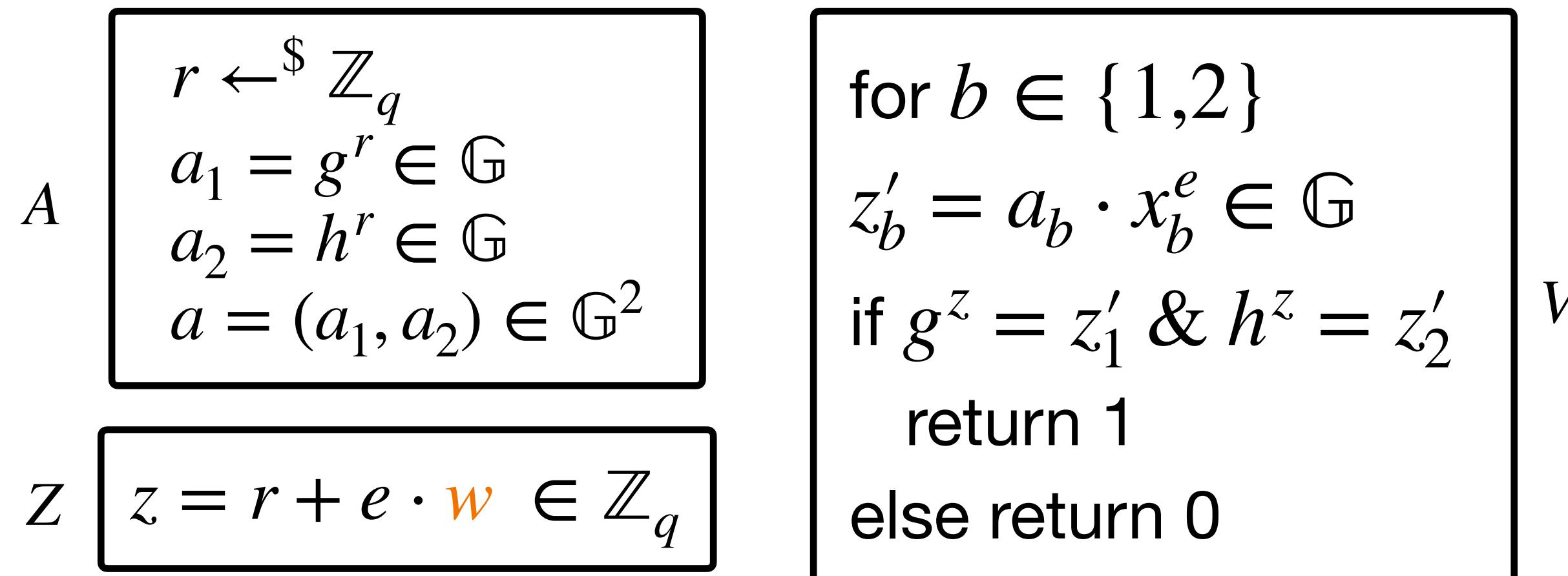
Chaum–Pedersen Σ -Protocol: Significance

$$R = \{ ((\mathbb{G}, q, g, h, x_1, x_2), w) \mid g, h \in \mathbb{G} \text{ & } x_1 = g^w \text{ & } x_2 = h^w \}$$



what can we use this for?

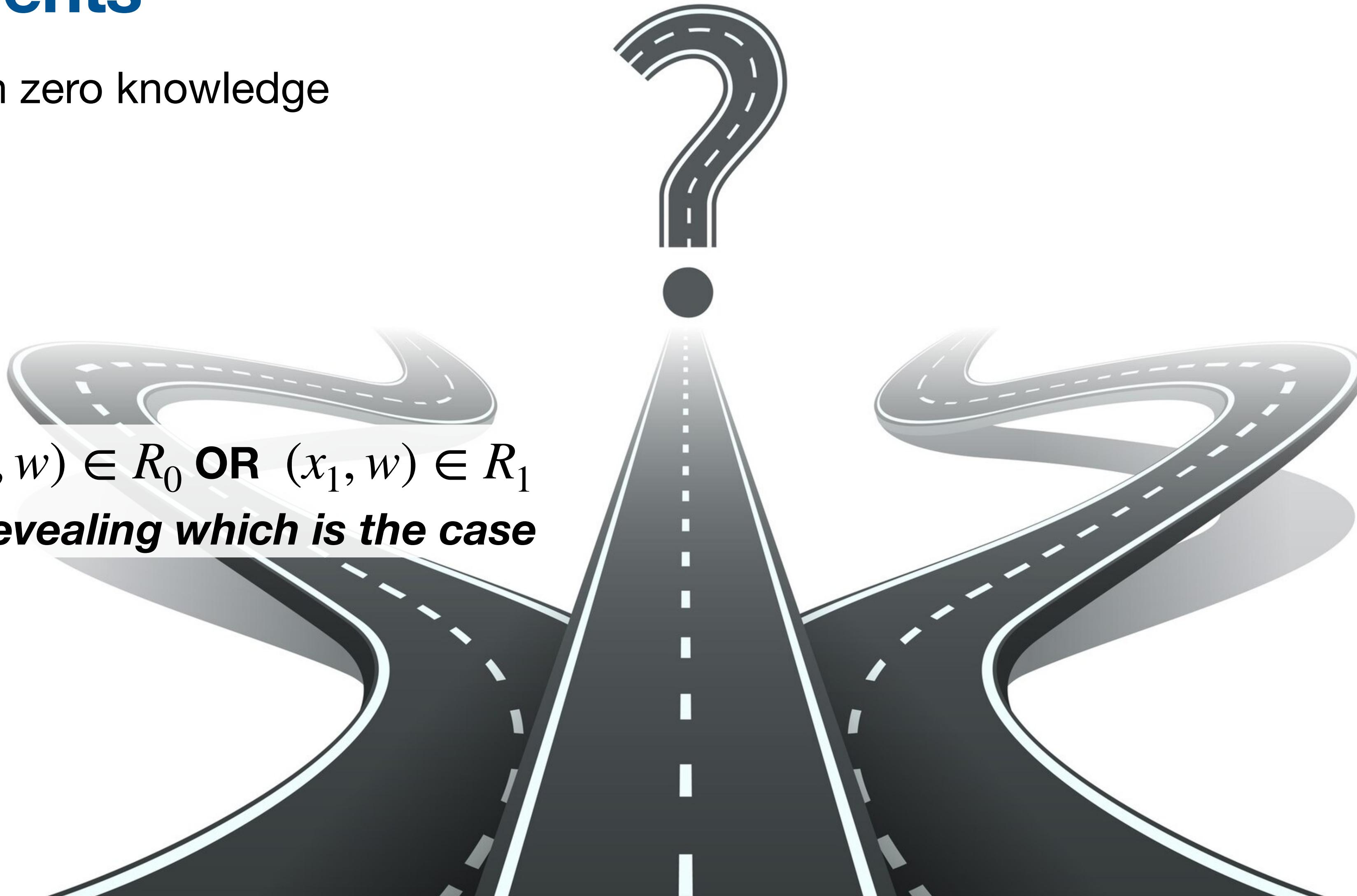
ZKPoK{ $w : x_1 = g^w$ and $x_2 = h^w$ }



Let $w = sk_A$ and $h = g^{sk_B}$ then this Σ -Protocol is a proof that (g, h, x_1, x_2) is a **DH Tuple!**

Proving OR of Statements

is a bit more complicated to do it in zero knowledge



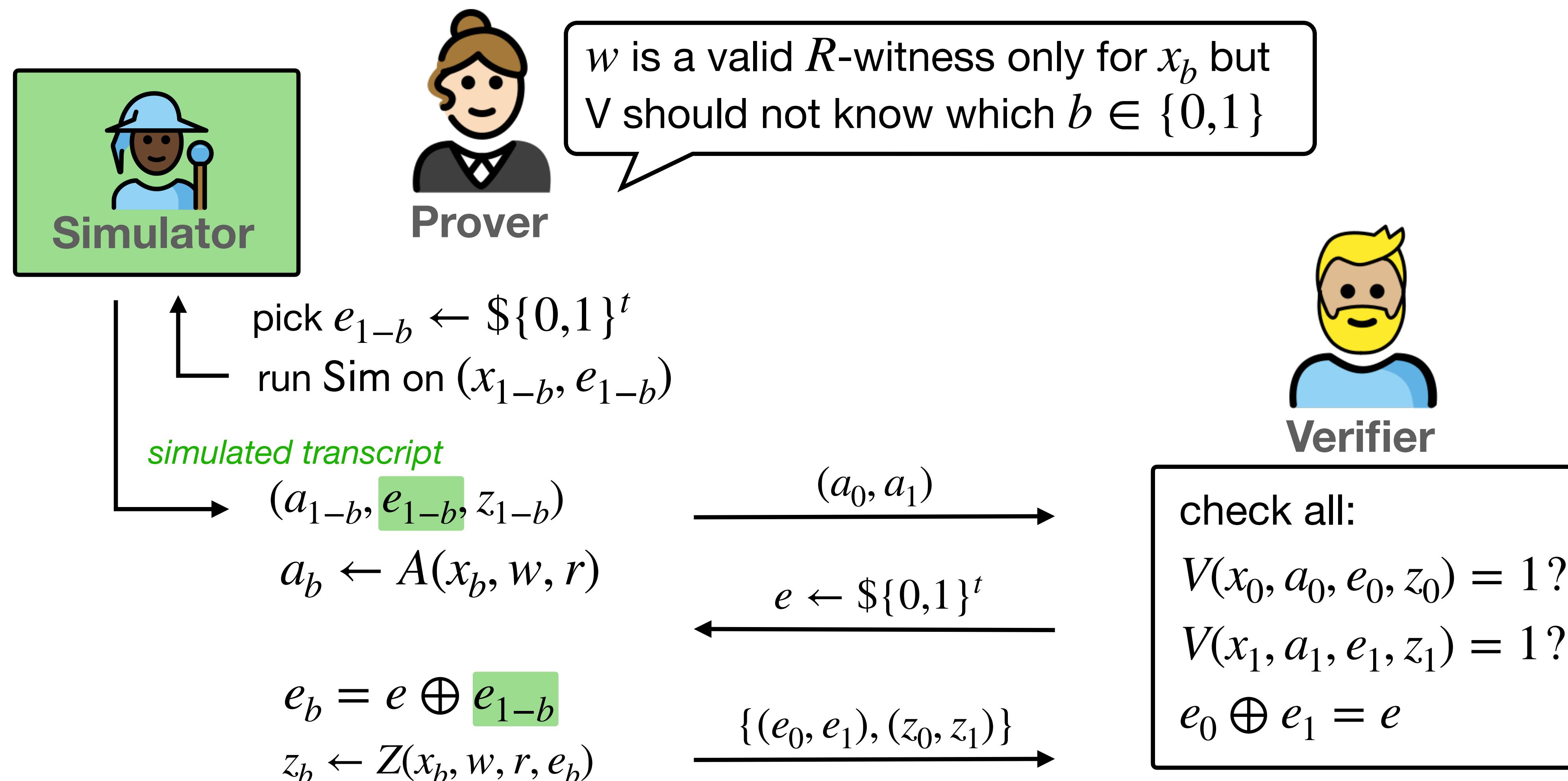
P wants to prove that: **either** $(x_0, w) \in R_0$ **OR** $(x_1, w) \in R_1$
ZK imposes to do this **without revealing which is the case**



The Trick : P completes the protocol for the instance x_b that is true and “fakes” a proof for the other statement by running the simulator (in a clever way)

Proving “OR” Statements

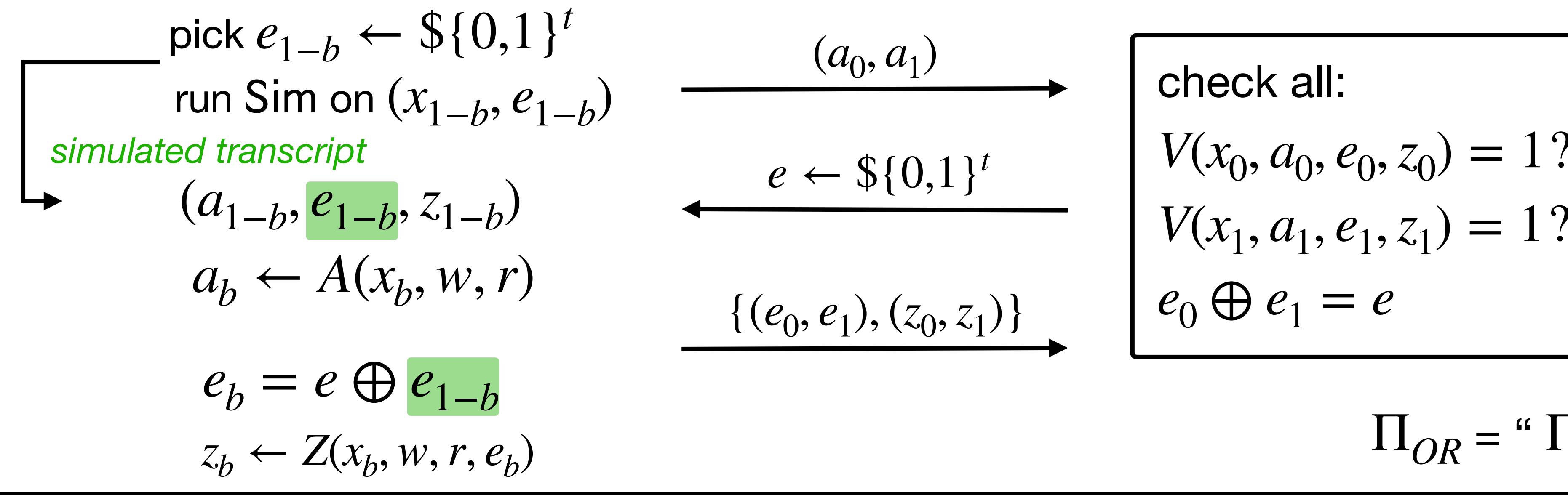
$ZKPoK\{w : R(x_0, w) = 1 \vee R(x_1, w) = 1\}$



The Trick : P completes the protocol for the instance x_b that is true and “fakes” a proof for the other statement by running the simulator (in a clever way)

Proving “OR” Statements

$ZKPoK\{w : R(x_0, w) = 1 \text{ or } R(x_1, w) = 1\}$



Completeness

follows from the completeness of $\Pi_0 = (A_{x_0}, Z_{x_0}, V_{x_0})$ & $\Pi_1 = (A_{x_1}, Z_{x_1}, V_{x_1})$

Special soundness: There exists a PPT algorithm \mathcal{E} (extractor) that given **any** x and any **pair of accepting transcripts** $(a, e, z), (a, e', z')$ for x , **with** $e \neq e'$, outputs w such that $(x, w) \in R$.

Lecture Agenda

Composing Σ Protocols

- In Parallel
- AND Proofs
- OR Proofs [Not in Exam]

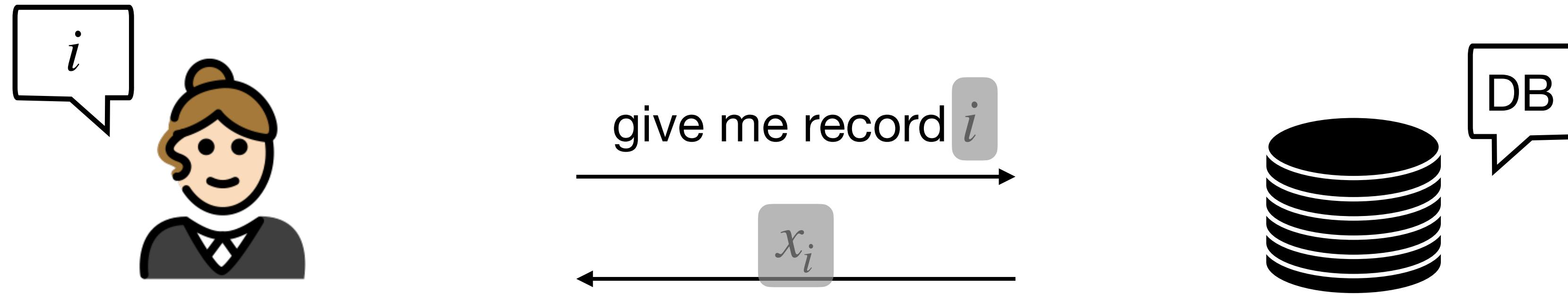
Private Information Retrieval (PIR)

- Problem Statement
- The Square Root PIR

Post Quantum Cryptography (PQC)

- The Lifespan of a Cryptosystem
- Classical VS Quantum Computers
- The Landscape of PQC
- Lamport Signature

Private Information Retrieval (PIR) - What Is It?

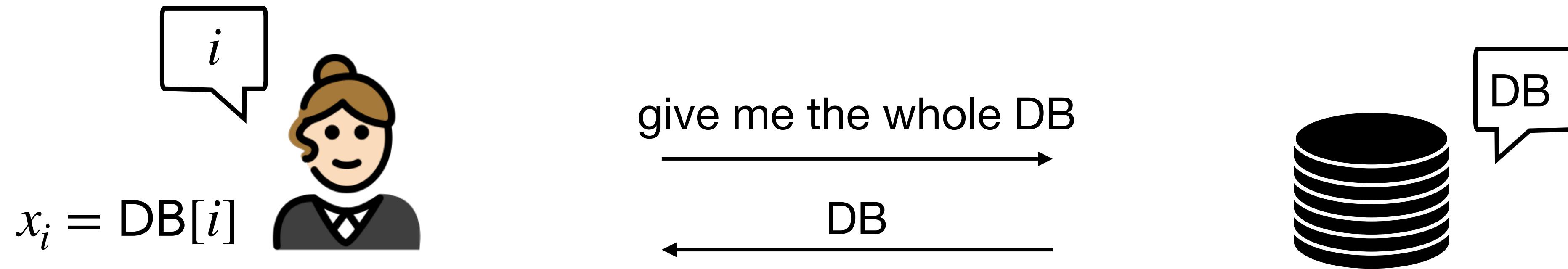


PIR Features: Alice learns (*retrieves information*) x_i , the server does not learn i (*private*)
[this is an example of Secure Multi-Party Computation!]

Many Applications: mobile contact discovery (PSI+OO.PIR); location privacy (DP+PIR);
anonymous communicationl (PIR-Tor); compromised credential checking (CIP-PIR);
...DNA search, donor matching, patent / internet domains registration...

A Trivial PIR

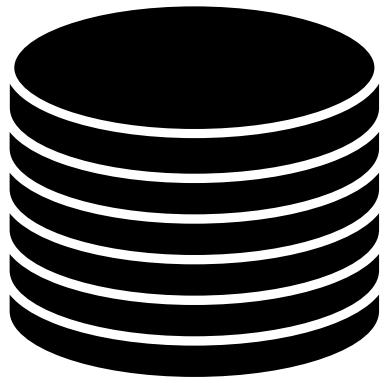
PIR Features: Alice learns (*retrieves information*) x_i , the server does not learn i (*private*)



🧐 Is this a PIR protocol?

The “Square-Root” PIR (Using Linearly Homomorphic Encryption)

Simplifying assumptions: the *data* are binary records, N is a square integer

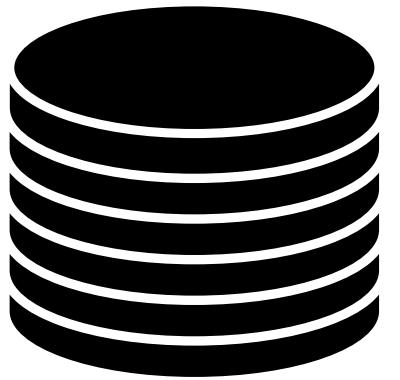


Step 0: Organise the database as a matrix

$$\text{DB} = \begin{array}{ccccccccccccccccccccccccc} \textcolor{pink}{\square} & \textcolor{pink}{\square} & \textcolor{pink}{\square} & \textcolor{pink}{\square} & \textcolor{pink}{\square} & \textcolor{blue}{\square} & \textcolor{blue}{\square} & \textcolor{blue}{\square} & \textcolor{blue}{\square} & \textcolor{teal}{\square} & \textcolor{teal}{\square} & \textcolor{teal}{\square} & \textcolor{teal}{\square} & \textcolor{orange}{\square} & \textcolor{orange}{\square} & \textcolor{orange}{\square} & \textcolor{orange}{\square} & \textcolor{orange}{\square} & \textcolor{red}{\square} & \textcolor{red}{\square} & \textcolor{red}{\square} & \textcolor{red}{\square} & \textcolor{red}{\square} & \end{array} \in \{0,1\}^N$$

The “Square-Root” PIR (Using Linearly Homomorphic Encryption)

Simplifying assumptions: the *data* are binary records, N is a square integer

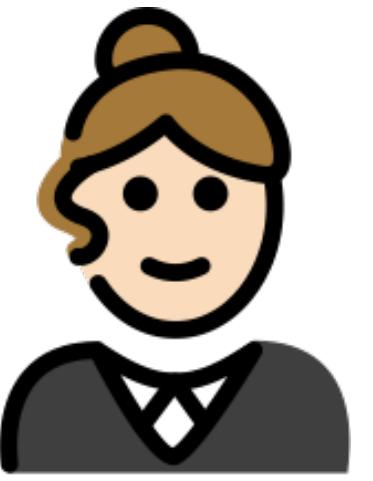


Step 0: Organise the database as a matrix

$$\text{DB} = \boxed{} \in \{0,1\}^N$$

$$M = \begin{matrix} \textcolor{pink}{\boxed{}} & \textcolor{pink}{\boxed{}} & \textcolor{pink}{\boxed{}} & \textcolor{pink}{\boxed{}} & \textcolor{pink}{\boxed{}} \\ \textcolor{blue}{\boxed{}} & \textcolor{blue}{\boxed{}} & \textcolor{blue}{\boxed{}} & \textcolor{blue}{\boxed{}} & \textcolor{blue}{\boxed{}} \\ \textcolor{teal}{\boxed{}} & \textcolor{teal}{\boxed{}} & \textcolor{teal}{\boxed{}} & \textcolor{teal}{\boxed{}} & \textcolor{teal}{\boxed{}} \\ \textcolor{orange}{\boxed{}} & \textcolor{orange}{\boxed{}} & \textcolor{orange}{\boxed{}} & \textcolor{orange}{\boxed{}} & \textcolor{orange}{\boxed{}} \\ \textcolor{darkred}{\boxed{}} & \textcolor{darkred}{\boxed{}} & \textcolor{darkred}{\boxed{}} & \textcolor{darkred}{\boxed{}} & \textcolor{darkred}{\boxed{}} \end{matrix} \in \{0,1\}^{\sqrt{N} \times \sqrt{N}}$$

The “Square-Root” PIR (Using Linearly Homomorphic Encryption)



Step 1: Send the encrypted query

Calculate on which column of M , the record i appears

Use LHE to encrypt (component-wise) the vector $c[j] = \text{Enc}(1)$ if $j = i \pmod{\sqrt{N}}$,
else $c[j] = \text{Enc}(0)$

Send the encrypted vector $\vec{c} =$

0
1
0
0
0

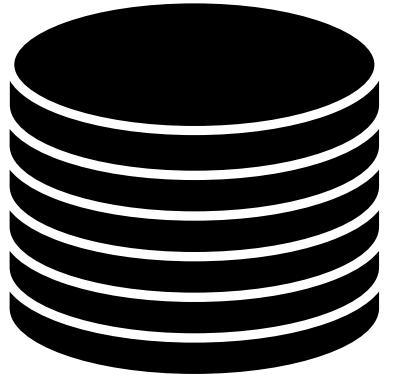
as a query

The entries of \vec{c} are encrypted

$M =$

pink	pink	pink	pink	pink
blue	blue	blue	blue	blue
green	green	green	green	green
yellow	yellow	yellow	yellow	yellow
brown	brown	brown	brown	brown

The “Square-Root” PIR (Using Linearly Homomorphic Encryption)



Step 2: Answer the query

Homomorphically evaluate $M \cdot \vec{c} = \vec{a}$

Return the vector \vec{a} as the answer

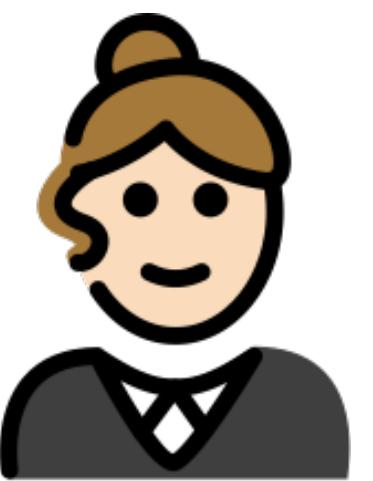
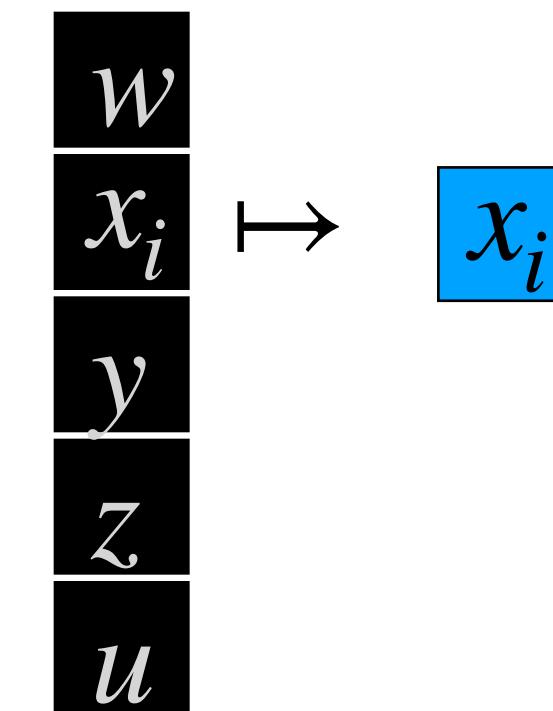
$$\begin{array}{c} i \\ \downarrow \\ \begin{array}{ccccc} \text{pink} & \text{pink} & \text{pink} & \text{pink} & \text{pink} \\ \text{blue} & \text{blue} & \text{blue} & \text{blue} & \text{blue} \\ \text{teal} & \text{teal} & \text{teal} & \text{teal} & \text{teal} \\ \text{orange} & \text{orange} & \text{orange} & \text{orange} & \text{orange} \\ \text{red} & \text{red} & \text{red} & \text{red} & \text{red} \end{array} \end{array} = \begin{array}{c} w \\ x_i \\ y \\ z \\ u \end{array}$$

Observe that
 $a[j] = \sum_{i=1}^{\sqrt{N}} x_{i+j \cdot N} \cdot \text{Enc}(0 \text{ or } 1)$

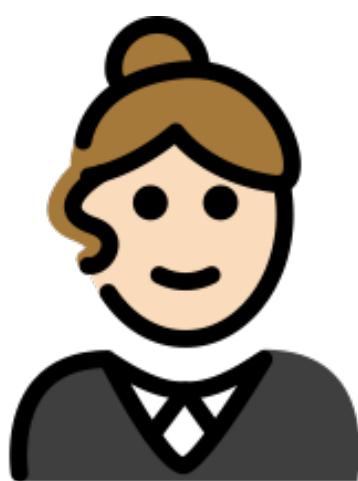
The “Square-Root” PIR (Using Linearly Homomorphic Encryption)

Step 3: Retrieve the record at position i

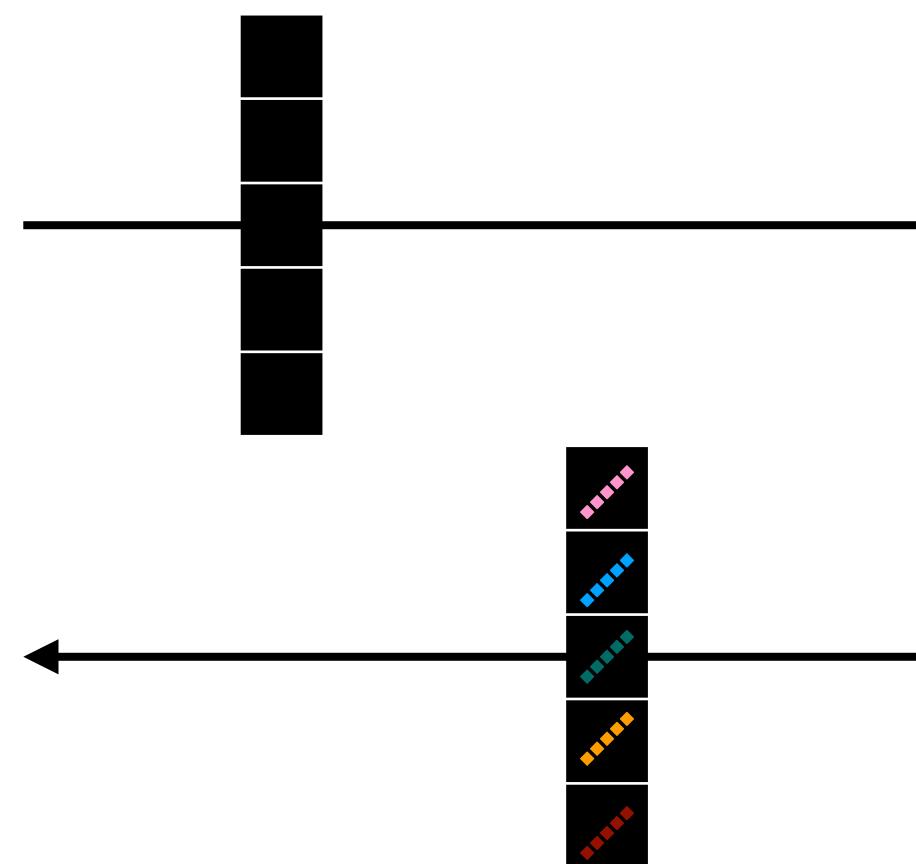
Decrypt $\vec{a}[i \bmod \sqrt{N}]$, this is the desired x_i !



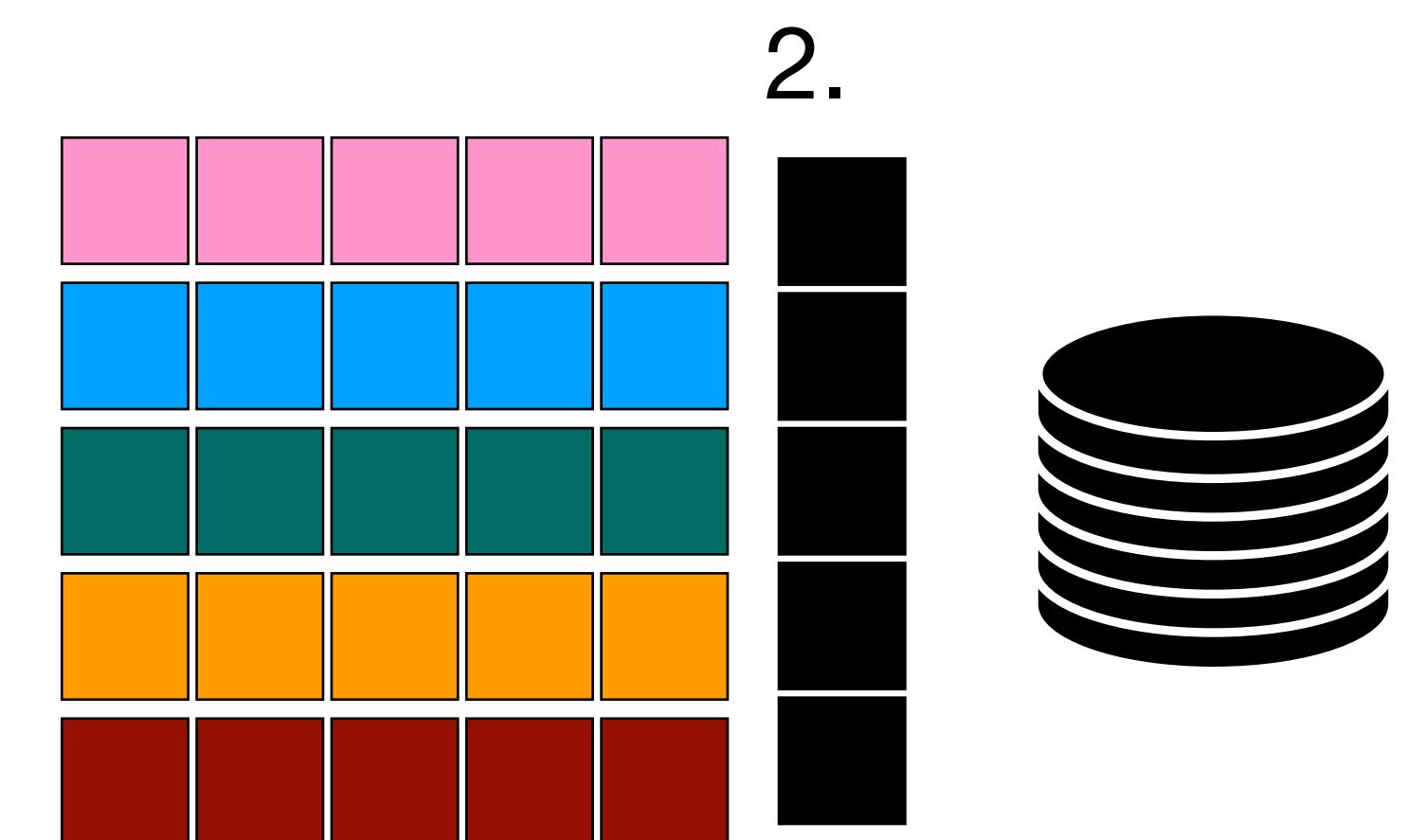
In Summary



1. $Enc(0 \text{ or } 1)$



3. $Dec(\text{target position}) \quad x_i$



Lecture Agenda

Composing Σ Protocols

- In Parallel
- AND Proofs
- OR Proofs [Not in Exam]

Private Information Retrieval (PIR)

- Problem Statement
- The Square Root PIR

Post Quantum Cryptography (PQC)

- The Lifespan of a Cryptosystem
- Classical VS Quantum Computers
- The Landscape of PQC
- Lamport Signature

The Life of the RSA Cryptosystem

RSA 1977

Rivest–Shamir–Adleman

new crypto system is proposed

web ~80-90s

cryptosystem is optimised for widespread

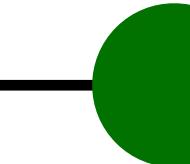
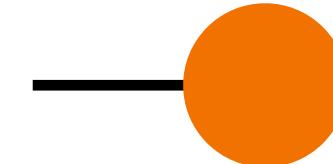
Shor 1994

Classical crypto broken by a PPT algorithm on a quantum computer

theoretic attack published

Powerful Large Scale Quantum Computers 2040(?)

the attack is practical



Timeline

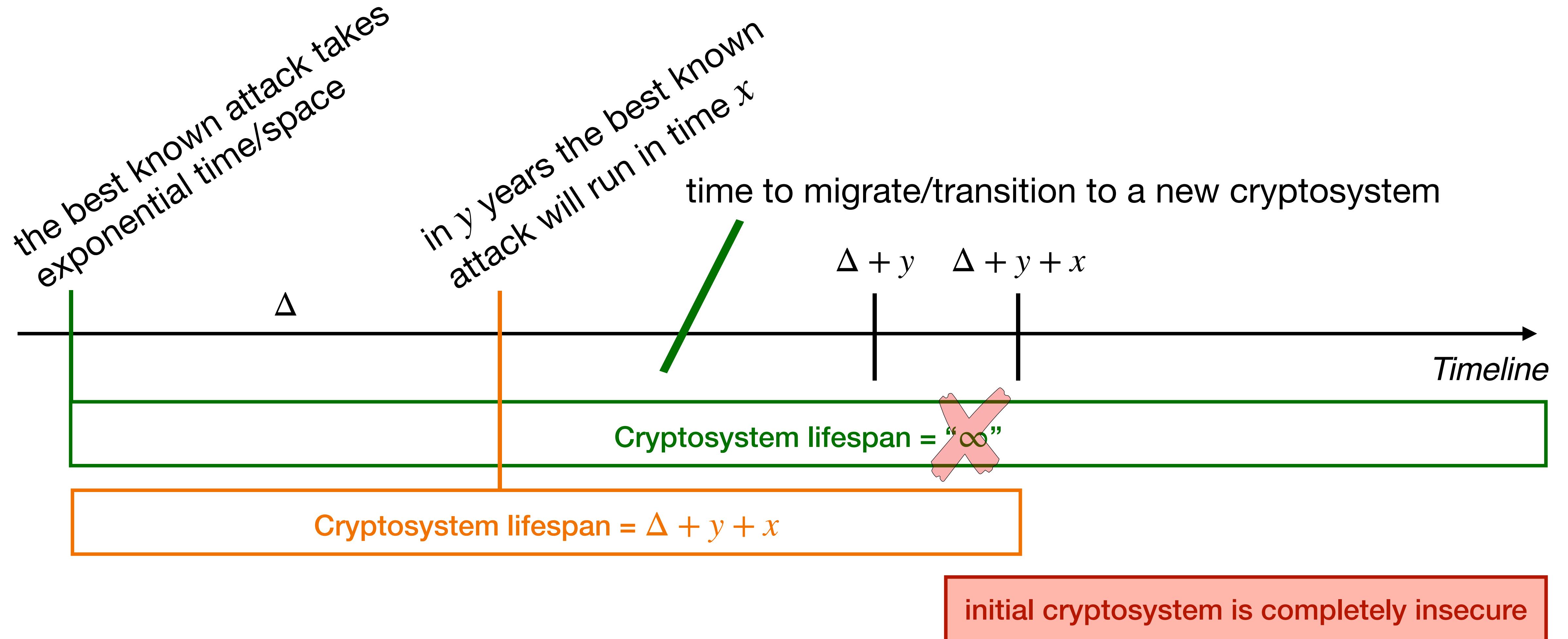
scrutiny by academics and researchers

continuous cryptanalysis, attacks, hacking

academics and researchers look for mitigations & improvements

Post-Quantum Cryptography

Cryptosystems Lifespan

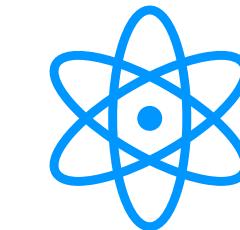


The Impact of Quantum Computing in Cryptography

Theorem 19.1 (Shor's Theorem)

The map that takes an integer m into its prime factorization is efficiently quantumly computable. Specifically, it can be computed using $O(\log^3 m)$ quantum gates.

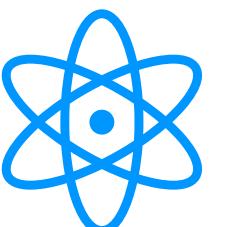
Informal translation: RSA is broken (both encryption and signature)!



Theorem 19.2 (Order Finding Algorithm)

There is a quantum polynomial time algorithm that given a multiplicative Abelian group \mathbb{G} and element $g \in \mathbb{G}$ computes the order of g in the group.

Informal translation: DH, ElGamal, ECDSA.... are all broken!



Yet, all of these crypto system are still widely used and we do not have clear evidence that quantum computers will soon reach the point where they can handle Shor's quantum algorithm.

Two Quantum Algorithms That Affect Cryptography

Polynomial-Time Algorithms for Prime Factorization
and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

Abstract

A digital computer is generally believed to be an efficient universal computing device; that is, it is believed able to simulate any physical computing device with an increase in computation time by at most a polynomial factor. This may not be true when quantum mechanics is taken into consideration. This paper considers factoring integers and finding discrete logarithms, two problems which are generally thought to be hard on a classical computer and which have been used as the basis of several proposed cryptosystems. Efficient randomized algorithms are given for these two problems on a hypothetical quantum computer. These algorithms take a number of steps polynomial in the input size, e.g., the number of digits of the integer to be factored.

affects most of the *classical*
public key cryptosystems

affects any symmetric key
cryptosystem, and hash functions



A fast quantum mechanical algorithm for database search

Lov K. Grover

Summary

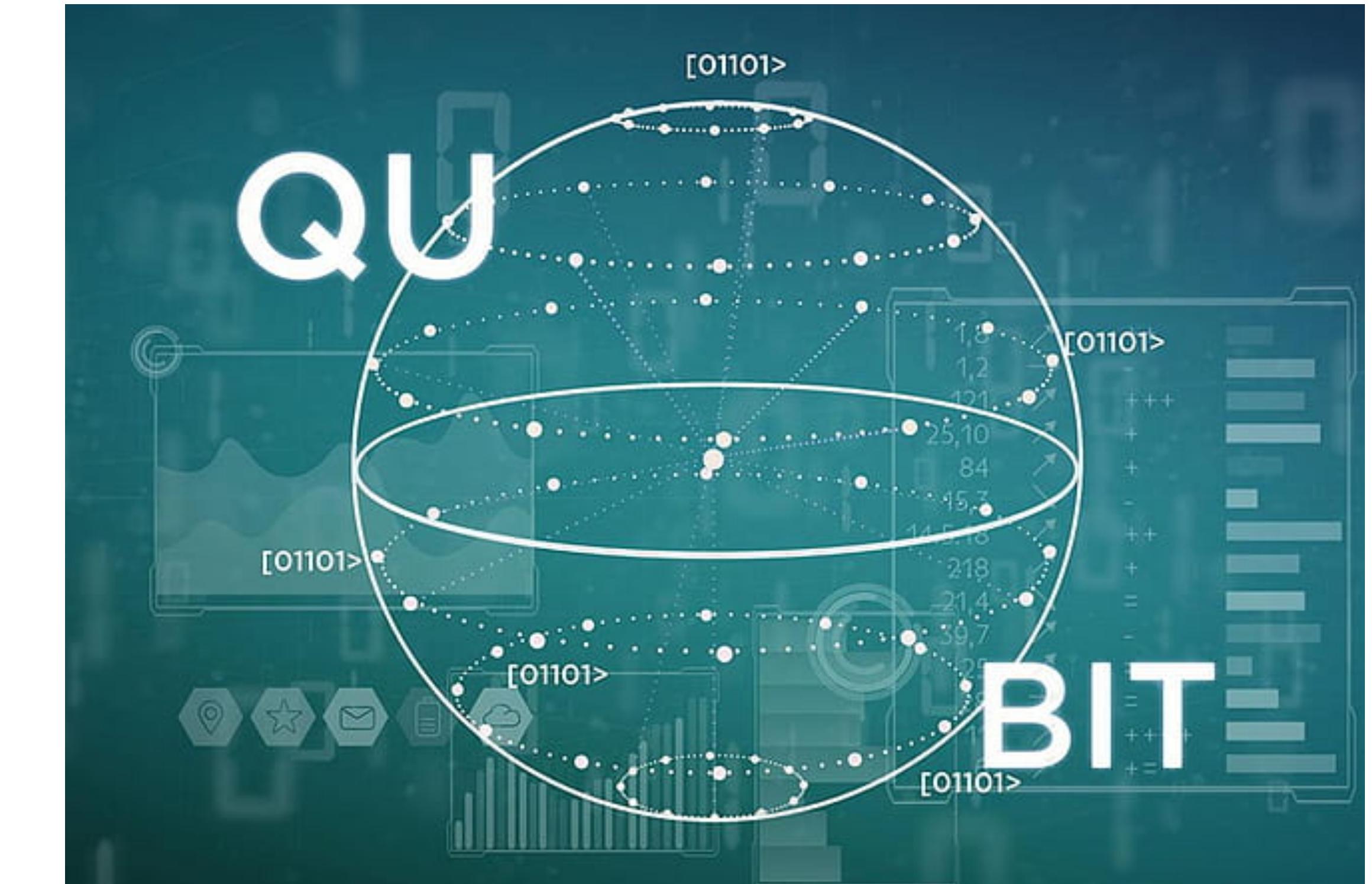
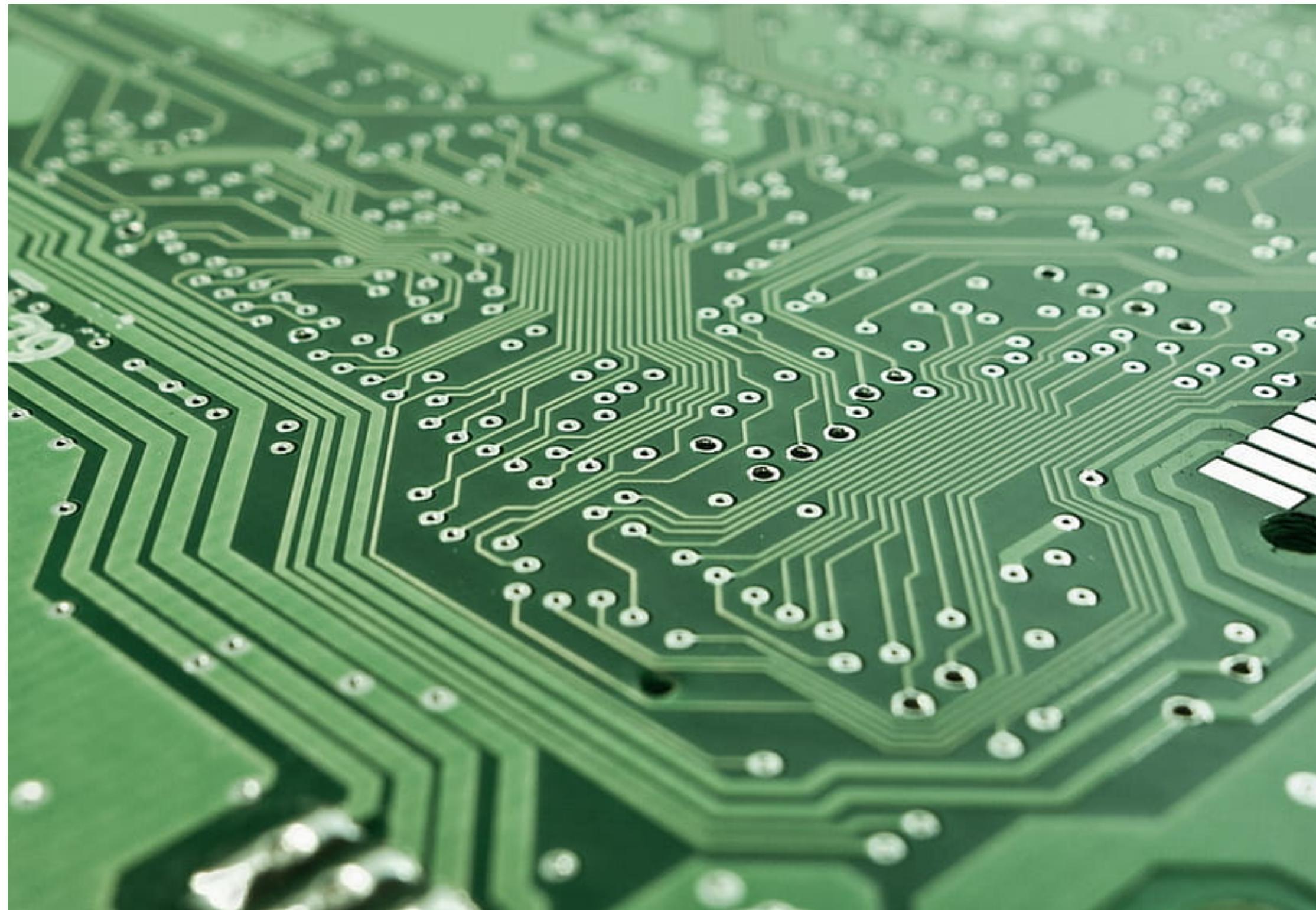
Imagine a phone directory containing N names arranged in completely random order. In order to find someone's phone number with a probability of $\frac{1}{2}$, any classical algorithm (whether deterministic or probabilistic) will need to look at a minimum of $\frac{N}{2}$ names. Quan-

tum mechanical systems can be in a superposition of states and simultaneously examine multiple names. By properly adjusting the phases of various operations, successful computations reinforce each other while others interfere randomly. As a result, the desired phone number can be obtained in only $O(\sqrt{N})$ steps. The algorithm is within a small constant factor of the fastest possible quantum mechanical algorithm.



How Long Will AES Remain “Secure” for?

classical vs quantum computing



Grover's algorithm runs a quantum brute force of AES keys in time $\sqrt{\text{classical}}$. A ‘simple’ mitigation that preserves security against quantum attackers is to **double the key length**.

Classical Vs Quantum Computers

The computational security of **classical** crypto relies on efficiently generating problems (using modern computers) that are intractable for **modern** computers.



SHE LOVES ME AND LOVES
ME NOT SIMULTANEOUSLY
WITH PROBABILITY
50% EACH...

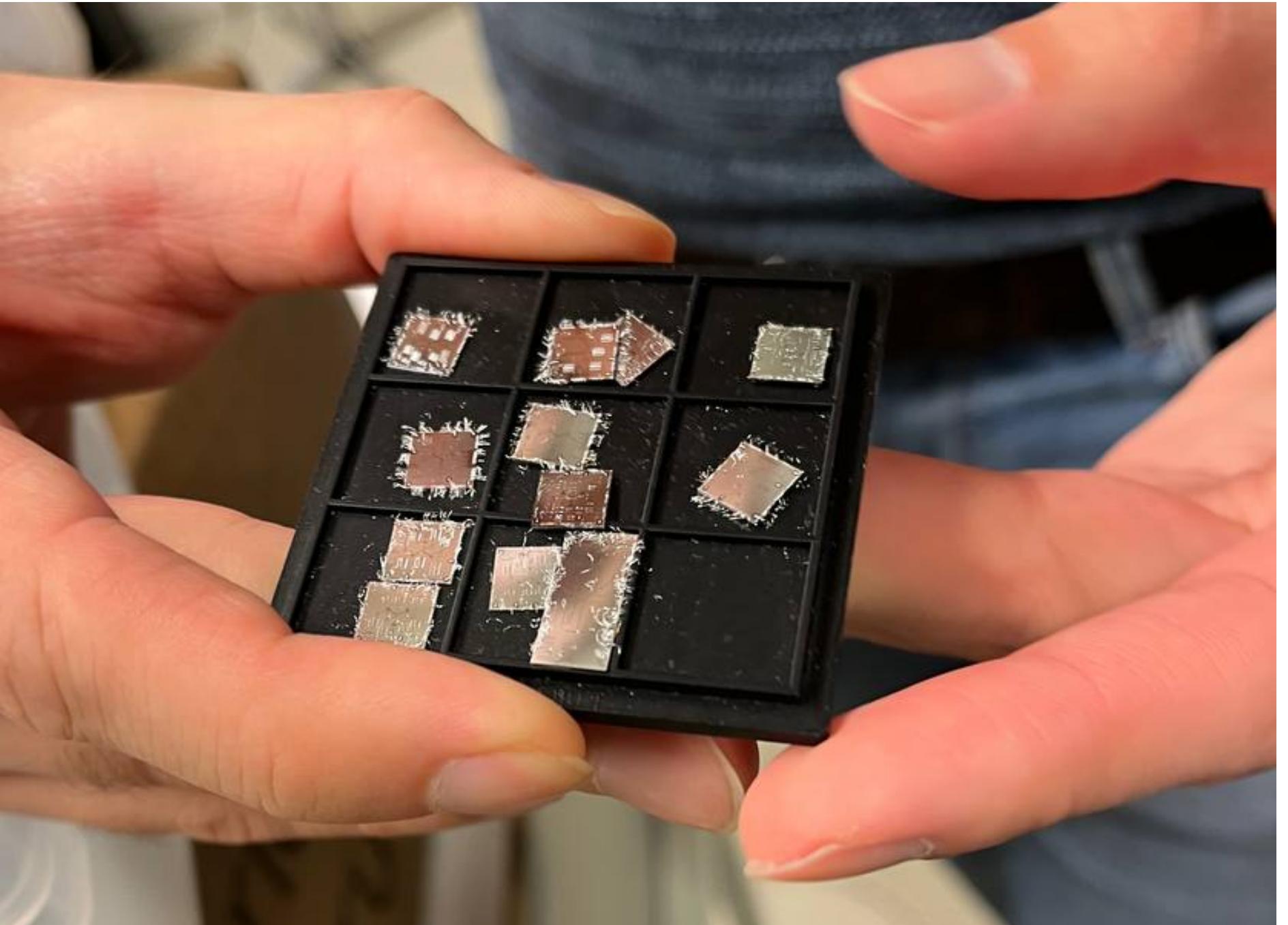


SCHRÖDINGER'S DAISY.



The computational security of **post-quantum** crypto relies on efficiently generating problems (using modern computers) that are intractable for **quantum** computers.

The State of Quantum Computers



@Chalmers: 25 qubits

Google: Sycamore chip, 54 qubits

IBM: 433 qubits

NSA: ?

200 seconds QC
10,000 years CC



How Many Qubits Are Needed To Break RSA?

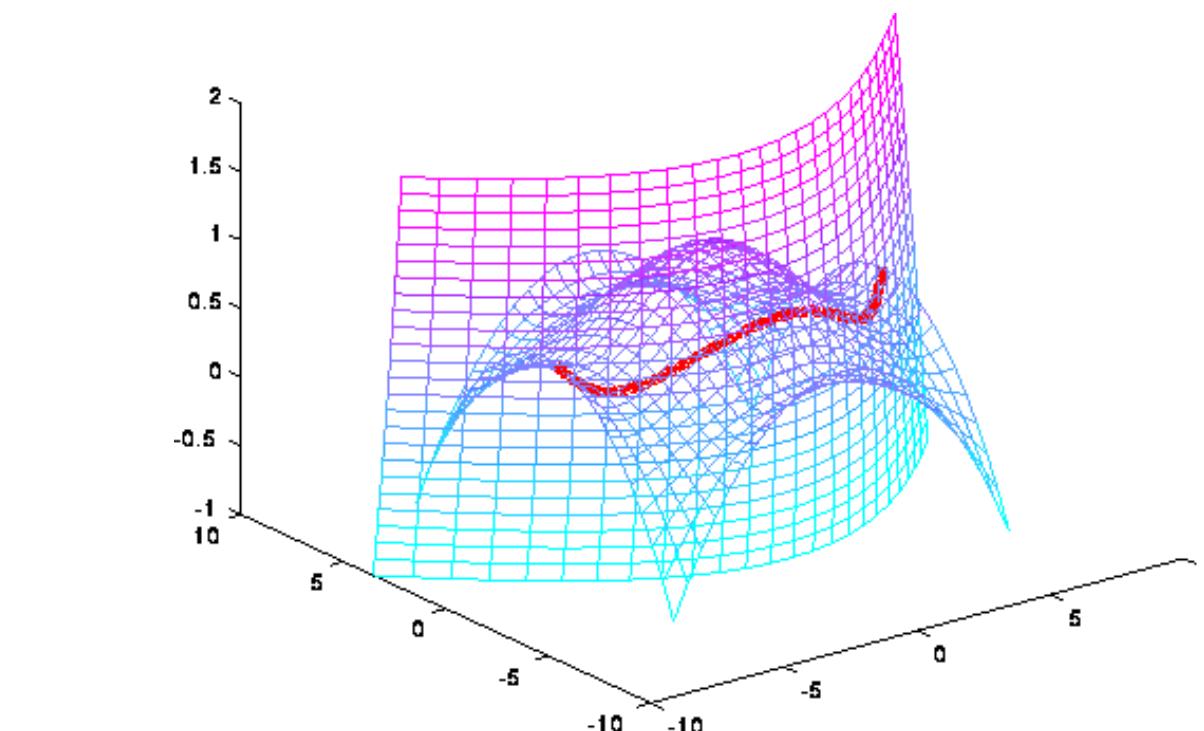
The current estimate is that breaking a 1,024-bit long RSA key requires a quantum computer with about 20 million qubits and about 8 hours of them running in superposition. [\[source\]](#)



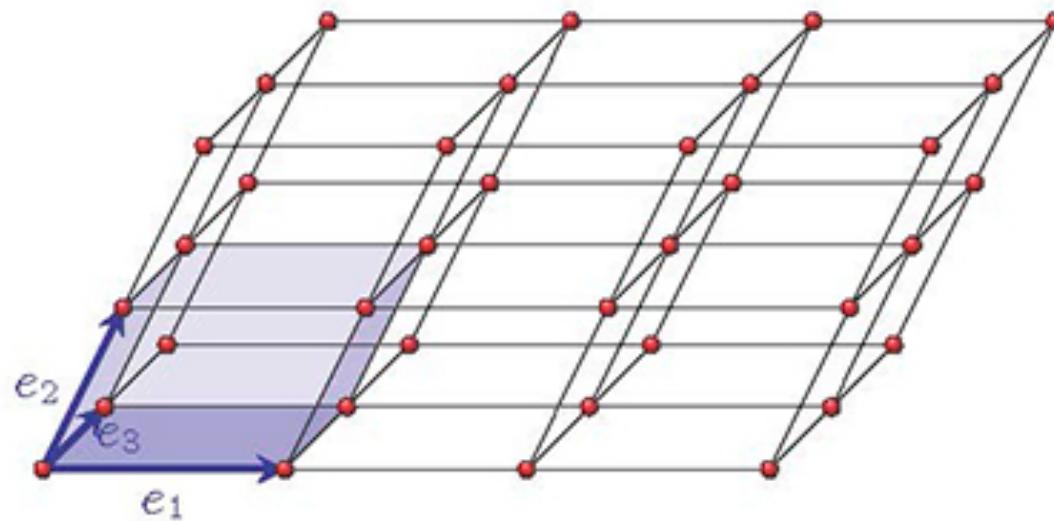
Post Quantum (PQ) Security

For more details check [Tanja Lange's lectures](#)

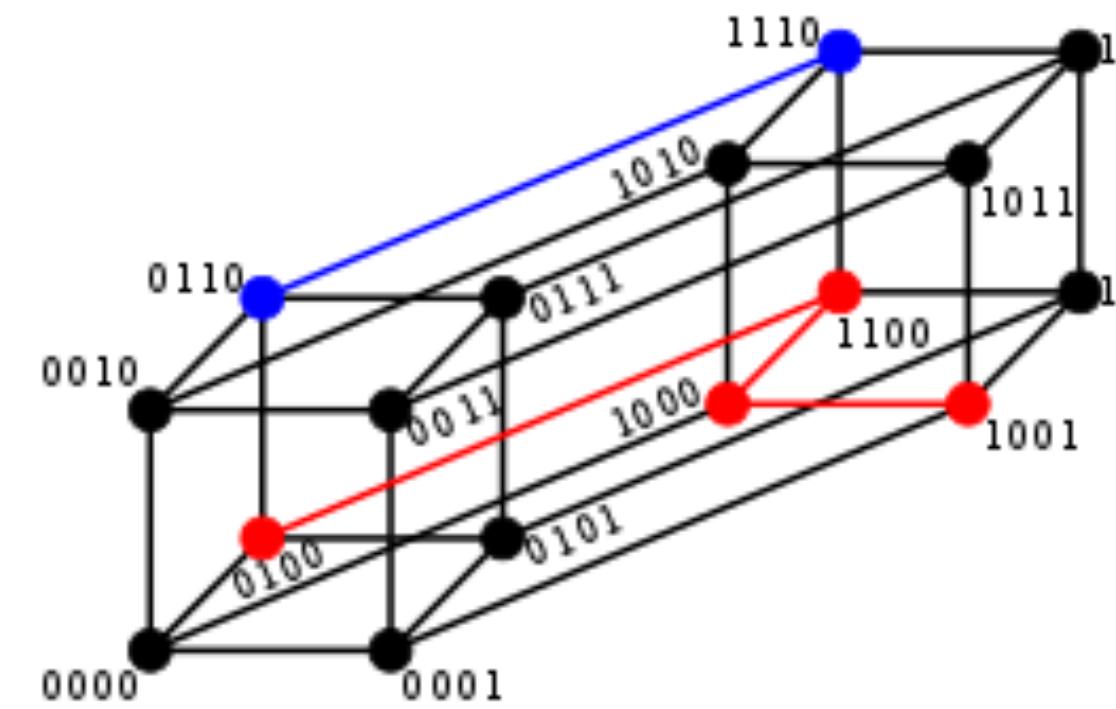
Multi Variate Quadratic Equations



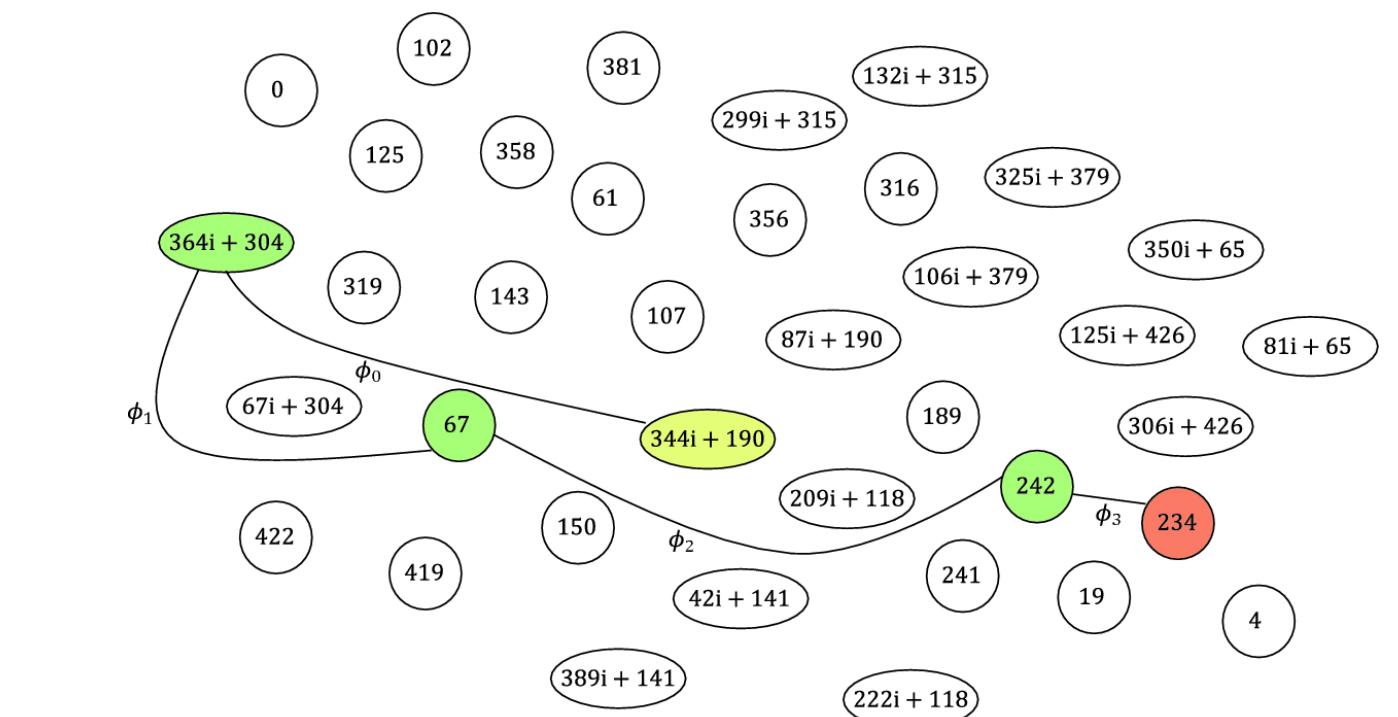
Lattices



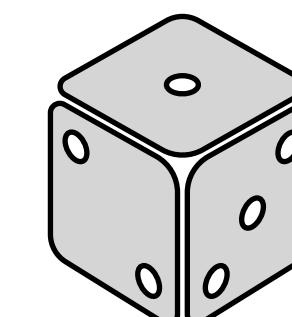
Codes



Isogenies



Hash Functions



Information Theoretic

Lamport Signature One-Time Signature (Hash-Based PQ)

KeyGen (1^n) \rightarrow (sk, pk)

$$\text{sk} = \begin{pmatrix} x_{1,0} & x_{2,0} & \dots & x_{v,0} \\ x_{1,1} & x_{2,1} & \dots & x_{v,1} \end{pmatrix} \leftarrow \$X^{2v}$$

$$\text{pk} = \begin{pmatrix} y_{1,0} & y_{2,0} & \dots & y_{v,0} \\ y_{1,1} & y_{2,1} & \dots & y_{v,1} \end{pmatrix} \text{ where } y_{i,b} = f(x_{i,b})$$

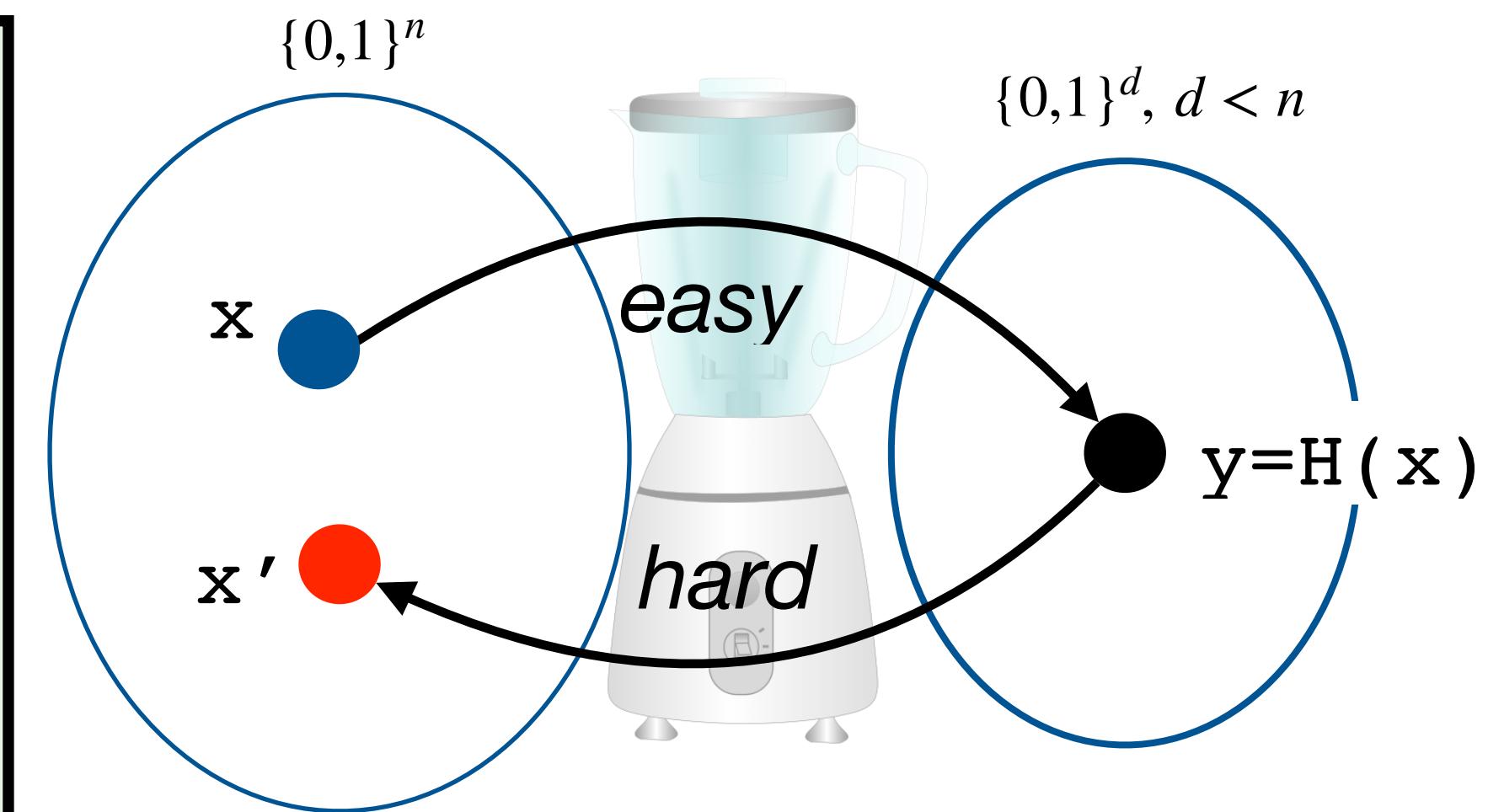
Sign (sk, m) \rightarrow σ

Take the bit decomposition of m

$$\sigma = (x_{1,m[1]}, x_{2,m[2]}, \dots, x_{v,m[v]})$$

Ver (pk, m, σ) \rightarrow {0,1}

Accept iff $f(\sigma_{i,m[i]}) = y_{i,m[i]}$ for all $i \in \{1,2,\dots,v\}$



$f: X \rightarrow Y$ is a one-way function and messages are in $m \in \{0,1\}^v$

Example

$$m = (1,0,\dots,0) \in \{0,1\}^v$$

NIST PQC Standardisation Effort

Post-Quantum Cryptography: Proposed Requirements and Evaluation Criteria

August 02, 2016

Announcing Request for Nominations for Public-Key Post-Quantum Cryptographic Algorithms

December 20, 2016

Announcing Approval of Three Federal Information Processing Standards (FIPS) for Post-Quantum Cryptography

August 13, 2024

ML-KEM
ML-DSA
SLH-DSA

NIST PQC Standardisation Effort ... Is Still Ongoing

Round 4 KEMs

(3 proposals based on Codes)

Round 2 Additional Signatures

(many proposals based on:
Codes,
Isogenies,
MPC-in-the-Head,
Multivariate Polynomial Equations,
Symmetric Crypto,
Lattice Isomorphism Problem)

